TPD/TOD Builders v1.05

User Manual

December 1, 2008

Copyright © 2003-2008
Serendipity Systems, Inc.
All Rights Reserved

Serendipity Systems, Inc.
P.O. Box 774507
Steamboat Springs, CO 80477

www.serendipsys.com
productinfo@serendipsys.com

PRODUCT INFORMATION:

TPD/TOD Builder Software (Referred to as “the Software” or “this Software”)
Copyright © 2003, Serendipity Systems, Inc., All rights reserved.
P.O. Box 774507
Steamboat Springs, CO 80477

1. Limited Liability

IF INSTALLED AND OPERATED AS REQUIRED, THE SOFTWARE SHOULD PERFORM AS DESCRIBED IN THE DOCUMENTATION ENCLOSED HEREWITH. ALL OTHER ASPECTS THE SOFTWARE IS PROVIDED "AS IS" WITHOUT ANY OTHER WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED.

YOU ARE NOT GRANTED ANY IMPLIED WARRANTIES OR MERCHANTABILITY AND/OR FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE SOFTWARE IS WITH YOU EXCEPT AS SET FORTH IN PARAGRAPH 5 BELOW. SHOULD THE SOFTWARE PROVE DEFECTIVE, YOU (AND NOT SSI OR ANY AUTHORIZED SSI DEALER) ASSUME THE ENTIRE RESPONSIBILITY AND COST OF ALL NECESSARY OR INCIDENTAL RESULTS PRODUCED, AS WELL AS ANY DAMAGES OF ANY KIND AND ANY SERVICE, REPAIR OR CORRECTION THAT MAY BE REQUIRED.

SOME STATES DO NOT ALLOW THE EXCLUSION OF IMPLIED WARRANTIES, SO THE ABOVE EXCLUSION MAY NOT APPLY TO YOU. THIS WARRANTY GIVES YOU SPECIFIC LEGAL RIGHTS AND YOU MAY ALSO HAVE OTHER RIGHTS WHICH VARY FROM STATE TO STATE.

Serendipity Systems, Inc. (SSI hereinafter) does not warrant that any of the functions contained in the Software will meet your requirements or that the operation of the Software will be uninterrupted or error free.

SSI warrants that the distribution media on which the Software is furnished to be free from defects in material or workmanship under its intended use, for a period of ninety (90) days from the date of delivery to you as evidenced by a copy of your receipt.

SSI shall not be responsible for any adverse effects caused by Acts of God or any other cause beyond SSI's reasonable control.

2. Limitation of Remedies

SSI's entire liability and your exclusive remedy shall be limited to the replacement within (30) days, for you (the original acquirer), of any distribution media not meeting SSI's Limited Warranty which is returned to SSI or any authorized dealer with a clearly legible copy of your receipt.

IN NO EVENT WILL SSI BE LIABLE TO YOU FOR ANY DAMAGES, INCLUDING ANY LOST PROFITS, LOST SAVINGS, LOST OPPORTUNITIES, OR OTHER INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OU T OF THE USE OR INABILITY TO USE SUCH SOFTWARE, EVEN IF SSI OR AN AUTHORIZED DEALER HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES, OR FOR ANY CLAIM BY ANY OTHER PARTY.

SOME STATES DO NOT ALLOW THE LIMITATION OR EXCLUSION OF LIABILITY FOR INCIDENTAL OR CONSEQUENTIAL DAMAGES SO THE ABOVE LIMITATION OR EXCLUSION MY NOT APPLY TO YOU.

ABOUT THIS DOCUMENT

This document contains operating instructions and development information for the Serendipity Systems' TPD and TOD Builders. This information is also contained in an online help file that is included with the product.

The information contained in this document is believed to be correct as of the date of publication. The authors assume no liability for any loss that might occur as a result of errors or omission of any information. Please report any errors to the authors for correction in later revisions of this document.

Table of Contents

1. INTRODUCTION 1

2. TPD BUILDER 3

2.1 TPD Hierarchy List 4
2.2 Status Bar 5
2.3 Toolbars 6
2.4 Menus 7
2.4.1 File Menu 7
2.4.2 Edit Menu 8
2.4.3 Project Menu 9
2.4.4 Options Menu 11
2.4.5 Help Menu 12
2.5 Dialogs and Windows 13
2.5.1 File Definition Dialog 13
2.5.2 Comment Window 14
2.5.3 TPD Builder Debug Log 15
2.6 Identification 16
2.7 Options 17
2.7.1 Logging & Printing 17
2.7.2 Display & Behavior 19
2.8 Setup Objects 20
2.9 Test Objects 21
2.9.1 Test Files 22
2.9.2 Prerequisites 23
2.9.3 Test Branching 24
2.9.4 Resolving Test Object Links 25
2.10 Action Objects 26
2.11 Other Files 27
2.12 Configuration 28
2.12.1 Configuration Data Dialog 29
2.12.2 Custom Label Dialog 33
2.12.3 Custom Test Types Dialog 34
2.13 Power Monitor 35
2.14 User Defined 36

3. TOD BUILDER 37

3.1 Verify Grid 38
3.1.1 Verify Type Column 39
3.1.2 Prefix Column 40
3.1.3 Units Column 41
3.2 Status Bar 42
3.3 Toolbar 42
3.4 Menus 43
3.4.1 File Menu 43
3.4.2 Edit Menu 44
3.4.3 Options Menu 46
3.4.4 Help Menu 47
3.5 Dialogs and Windows 48
3.5.1 Find Dialog 48
3.5.2 Fill Column Dialog 49

3.5.3 TOD Builder Debug Log 50

4. OPERATIONAL NOTES 51

4.1 TEST CRITERIA 51
4.2 TEST METHODS 52

5. BUILDER FILES 53

5.1 TOD BUILDER CONFIGURATION FILE 53
5.2 XML EXPORT FILE 54
5.3 TEST OBJECT DEFINITION FILE 56
5.3.1 Test Object Definition Format 57
5.3.2 TOD Identification 58
5.3.3 TOD Verifies 59
5.4 TEST PROGRAM DEFINITION FILE 60

6. APPENDIX A - FORMATTING MASK 61

7. APPENDIX B - VERSION CHANGES 63

8. SERENDIPITY SYSTEMS, INC. 67

1. Introduction

This describes the operation of the Test Program Definition (TPD) Builder and the Test Object Definition (TOD) Builder, developed by Serendipity Systems, Inc. (SSI). TPD and TOD files are used to define the behavior of the SSI Test Executive.

The SSI Test Executive is an application that coordinates the selection, execution and reporting of ATE-based test programs. A fundamental aspect of the Test Executive is the comparison of measured values to defined test limits. These comparisons, or verifies, are key to test result reporting and trend analysis. The test limits can be contained within a test program or defined externally in a Test Object Definition file. Test operation, sequencing, reporting, and behavior is controlled by a Test Program Definition file.

The TPD Builder provides a high-level user interface for creating, editing and saving TPD files. This allows a test developer to focus on test set functionality rather than TPD file syntax. In addition, the TPD Builder has tools for automatically updating file information and validating a TPD file's contents.

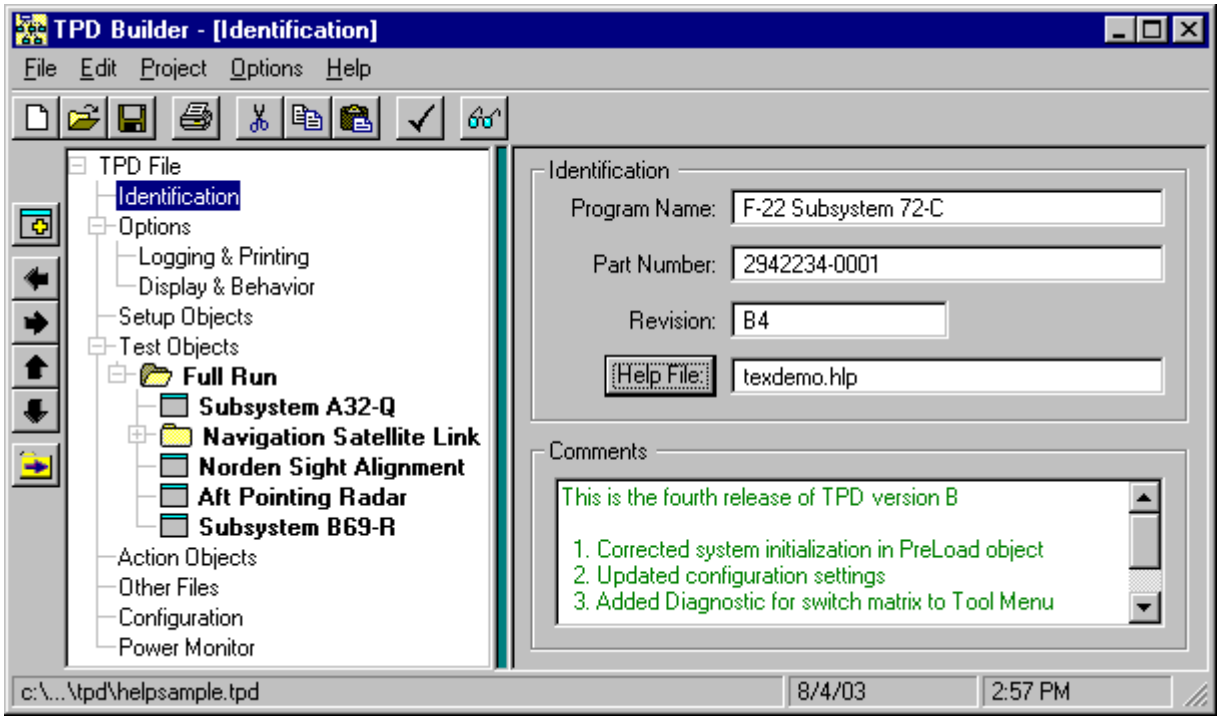
The TOD Builder is an application that aids the development of TOD files. TOD files are used to isolate test limits from the program that performs the measurements. This allows the limits to be adjusted without rebuilding the executable. This is extremely valuable when adapting tests for engineering changes or when using different limits for production versus environmental test. Limits for several different test programs can be stored in one TOD file. This can make tracking, installing and updating simpler. The line-oriented format of a TOD file allows it to be easily incorporated into a source control system (e.g. Visual SourceSafe).

The simple format of a TOD file allows it to be manually edited with any plain text editor (e.g. Notepad). When managing hundreds, or thousands, of test limits, manual editing can become tedious and error-prone. Consequently, Serendipity Systems has developed the TOD Builder for creating, changing, viewing and validating TOD files. This tool has a spreadsheet style interface that provides simple entry, error visibility, embedded comments, selection lists, automated fill and cut/paste operations. In addition, the TOD Builder supports XML import and export.

Familiarity with the Test Executive, and TPD/TOD files, is helpful when using these applications. For a more detailed understanding of many of the operational concepts, refer to the Test Executive manual or help file.

2. TPD Builder

The TPD Builder has an easy-to-use interface for creating and editing a TPD file. With it, a test developer can load a TPD file, create and modify a test hierarchy, edit operational parameters, validate contents and save changes. This application greatly simplifies the creation of TPD files for use by the SSI Test Executive.

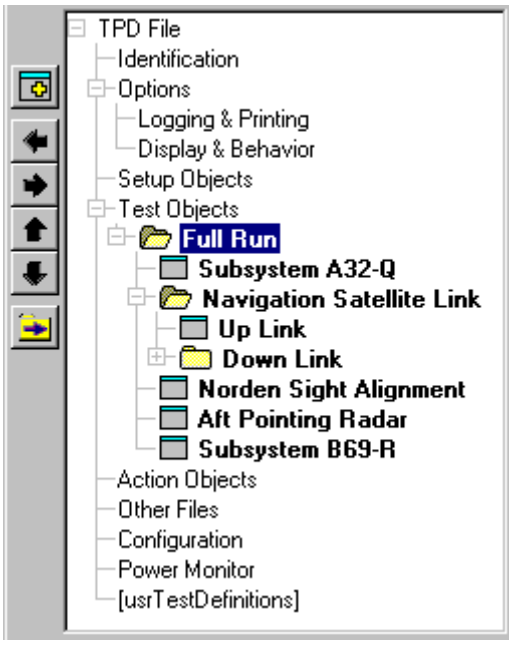


TPD loading and saving are controlled from the File Menu or toolbar. The hierarchy list is edited via the Project Menu or sidebar. Standard editing commands on the Edit Menu include cut, copy, paste, insert and delete. Additional commands available via the Options Menu include validate and show paragraph numbers. The Debug Log window is provided to view errors and warnings resulting from validation. The Help Menu accesses online help and version information.

A through understanding of the Test Executive functionality, and TPD file elements, is quite beneficial when using the TPD Builder. For a more detailed understanding of many of the TPD settings, refer to the *Test Program Definition File* section of the Test Executive manual or help file.

2.1 TPD Hierarchy List

The TPD Builder Hierarchy List displays the sections, and test hierarchy, of the currently loaded TPD file. Selecting an item in the list, via a left mouse click, displays its properties in a window to the immediate right. The items in the list that correspond to the sections in a TPD file (e.g. Identification, Options, etc.) are fixed and cannot be moved or deleted. The test object portion of the list displays the structure of the test list that is displayed in the Test Executive. These items can be added, deleted, or moved via the Project Menu, right-click pop-up menu, and sidebar (shown below).



Items that have no direct properties (e.g. TPD File, Options, Test Objects), provide windows for entering comments. Within these windows, text is colored green to indicate that it will be stored as comments in the TPD file. It is not necessary to use the TPD comment characters (***//***) in these windows.

The TPD file format permits user defined sections to be added to a TPD file. These are identified by a section header that is prefixed by "**usr**" (e.g. [usrTestData]). When such sections are present, the TPD Builder adds them to the bottom of the hierarchy list. Selecting a user defined section causes its contents to be displayed in a text window to the immediate right.

Double-click an item's text, or icon, to collapse or expand its children. A single click on the plus/minus (+/-) image also collapses or expands an item's children.

2.2 Status Bar

The TPD Builder status bar displays the path and name of the currently loaded TPD file.

2.3 Toolbars

The toolbars on the TPD Builder provide quick access to commonly used commands. All of the buttons on the toolbars have corresponding menu entries. Many of them also have shortcut keys that perform the same command. The shortcut keys are shown to the right of their menu entries. Tooltips are displayed when the mouse cursor is held over a toolbar button for approximately two seconds. A tooltip box appears temporarily to identify the button's function.

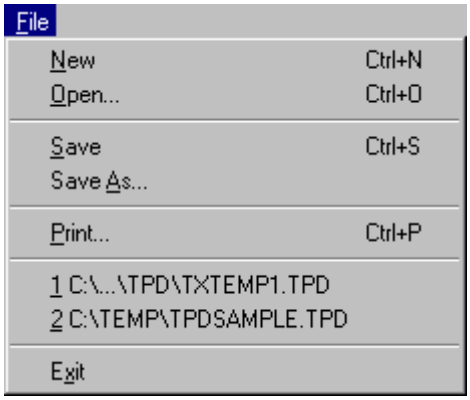
The toolbar at the top of the TPD Builder window contains a typical set of file, print and edit commands. In addition, the checkmark button validates the currently loaded TPD file, and the eyeglasses button activates the Debug Log window.

The toolbar on the left side of the TPD Builder window has buttons that correspond to items on the Project Menu. These facilitate building and modifying the test objects portion of the TPD Hierarchy List. This toolbar is only enabled when a test object, or test group, is selected. Note that moving a test group also moves the test objects that are indented underneath it. When a test object moves left or right, it becomes a test group if there are test objects indented immediately under it. The lowermost **Slip Right** button allows you to move a test group to the right without its children. This can restore a test object that has inadvertently become a test group by moving too far to the left.

2.4 Menus

2.4.1 File Menu

The TPD Builder File Menu supports loading and saving TPD files. When a TPD file is loaded, it is syntax checked and the results are written to the Debug Log window. A file history list permits quick reloading of recently accessed TPD files. Several toolbar buttons and shortcut keys are provided to more easily initiate these operations.

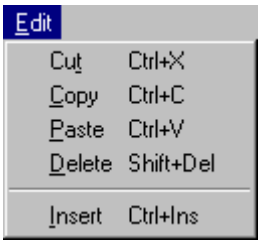


Menu Option	Description
New...	Opens a file browser for naming and placing a new TPD file. A set of default entries are placed in the TPD Hierarchy List.
Open...	Opens a file browser for selecting a TPD file. The selected file is loaded and displayed in the TPD Hierarchy List.
Save	Updates the TPD file with the latest editing changes.
Save As…	Creates a new TPD file with the latest editing changes. It then becomes the current file.
Print…	Prints a screen capture of the application on the selected printer.
File History	This provides a quick selection from the last four TPD files opened. The list is updated each time a new file is opened.
Exit	Closes the TPD Builder. The current window positions and control selections are saved in an initialization (INI) file.

2.4.2 Edit Menu

The TPD Builder Edit Menu is used to cut, copy, paste and delete text in many of the data entry windows. The delete and insert commands allow adding or removing individual rows from a file entry grid.

Several toolbar buttons and shortcut keys are provided to more easily initiate these operations. This menu is also accessible as a pop-up menu (right mouse click on the control). Individual fields, and controls on other windows, always support the shortcut keys for cut (**Ctrl+X**), copy (**Ctrl+C**) and paste (**Ctrl+V**).



Menu Option	Description
Cut	Copy the selected text to the system Clipboard and remove the text. Also achieved via **Ctrl+X**.
Copy	Copy the contents of the selected text to the system Clipboard. Also achieved via **Ctrl+C**.
Paste	Insert the contents of the system Clipboard into a text entry control. Also achieved via **Ctrl+V**.
Delete	Delete the selected text. Also used to delete a selected row when editing a file entry grid such as Setup Objects. The row below is moved up and selected. This allows rapid deletion of multiple rows when used with **Shift+Del**.
Insert	Insert a minimally initialized row, above the selected one, in a file entry grid. Also achieved via **Ctrl+Ins**.

2.4.3 Project Menu

The TPD Builder Project Menu is used to manipulate the items in the test object portion of the TPD Hierarchy List. This menu allows test objects and test groups to be added, deleted, renamed and moved within the hierarchy. The move commands are best performed with the sidebar buttons. This menu is also accessible as a pop-up menu (right mouse click on the control).

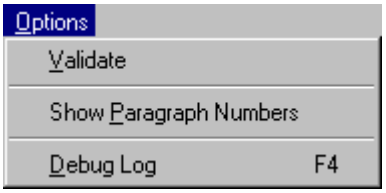
A test group is created when one or more test objects are indented underneath it. A test group does not include test files (i.e. TOD and an executable). Moving left and right can cause a test object to become a test group (use slip right to move a test group right without including its children). When a test group moves, its children move with it.

> **Note: Deleting a test group also deletes all test objects that are indented underneath it.**

Menu Option	Description
Add…	Activates a submenu for adding a test object entry below the current selection. The new entry is created with a default name that is selected for immediate editing.
Delete	Delete the selected test object or test group entry. Deleting a test group also deletes the test objects indented underneath it.
Rename	Allows changing the name of the selected test object or test group. Can also be done from within the Test Object window.
Move Left	Move the selected test object, or test group, to the left in the hierarchy.
Move Right	Move the selected test object, or test group, to the right in the hierarchy.
Move Up	Move the selected test object, or test group, up in the hierarchy.
Move Down	Move the selected test object, or test group, down in the hierarchy.
Slip Right	Move the selected test object, or test group, to the right in the hierarchy. Test groups "slip" right without changing the test objects beneath them.

2.4.4 Options Menu

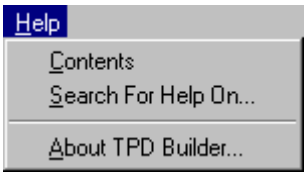
The TPD Builder Options Menu is used to activate special operations for managing the contents of a TPD file. The Validate command checks TPD syntax, contents, file existence/versioning, and general usage. Validation errors are displayed in the Debug Log window. Visible paragraph numbering is helpful when targeting test branching, or prerequisites, in a large hierarchy. Validate and the Debug Log command are also available on the toolbar.



Menu Option	Description
Validate	Scan TPD contents and report warnings and errors in the Debug Log window.
Show Paragraph Numbers	Toggles paragraph numbering for the items in the test object portion of the hierarchy list.
Debug Log	Allows the user to toggle between the Debug Log window and the TPD Builder. Function key **F4** also performs this task.

2.4.5 Help Menu

The TPD Builder Help Menu provides access to the online help file and version information. The help file is most easily accessed by pressing the **F1** function key. The F1 key also works for accessing help for dialog windows such as File Definition and Configuration Data.



Menu Option	Description
Contents	Displays the Contents page of the help file. Function key **F1** also performs this task.
Search for Help On…	Displays the Search index of the help file. Use this to locate specific information in the help file.
About TPD Builder…	Displays a dialog containing version information, a serial number and copyright notice.

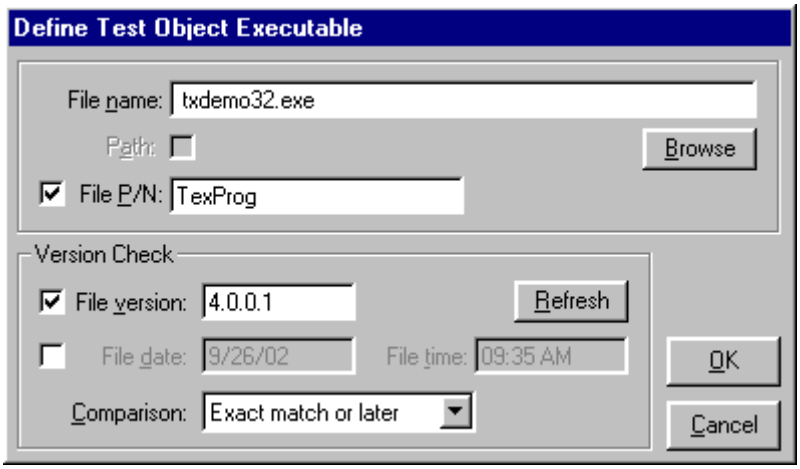
Note: Vista users may be prompted to download the Windows Help program (WinHlp32.exe) from the Microsoft support website in order to display files in the Windows Help format (*.hlp).

2.5 Dialogs and Windows

2.5.1 File Definition Dialog

The File Definition dialog is used, within the TPD Builder, to enter, select, update and modify information associated with a file reference. Most file references, in a TPD, also support part number and version checking. When a file is selected, via browsing, the part number and version information is automatically displayed in this dialog. Checkboxes control which of these items will be used by the Test Executive to validate the file's presence. The **Comparison** dropdown box selects the type of versioning check (none, exact, exact or later). The **Refresh** button locates the specified file and updates the version information. This is particularly useful when preparing a TPD for final release.

> **Note: TOD files and test object executables cannot have directory paths, so the Path checkbox is always disabled. To improve TPD portability, avoid using directory paths, for file definitions, wherever possible.**



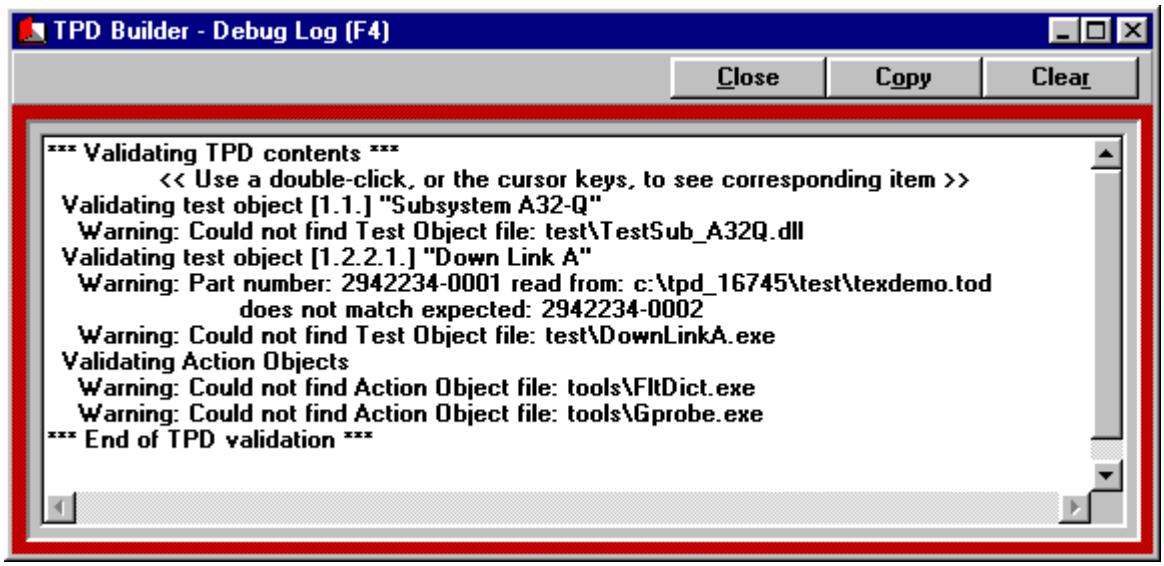
For a more detailed understanding of file verification, and version checking, refer to the Test Executive manual, or help file, sections: *TPD Directory Structure*, *File Existence Checking* and *Version Checking*.

2.5.2 Comment Window

The TPD Hierarchy List has several items that have no direct relationship to TPD file entries. These items include: **TPD File**, **Options** and **Test Objects**. Consequently, when selected, a text window is provided for entering comments that will be embedded in the TPD file. Within this window, text is colored green to indicate that it will be stored as a comment in the TPD file. It is not necessary to use the TPD comment characters (***//***) in this window.

2.5.3 TPD Builder Debug Log

The Debug Log window displays warnings and errors that occur during TPD Builder operations. This includes problems encountered when loading, saving or validating TPD files. The Debug Log window is activated via the Options Menu. Function key **F4** toggles between the Debug Log and the TPD Builder window. Scroll bars and key commands (**PgUp**, **PgDn**, **Home**, etc.) are used to move around the window. Push buttons are provided to copy the selected contents to the Windows clipboard or to clear the display.



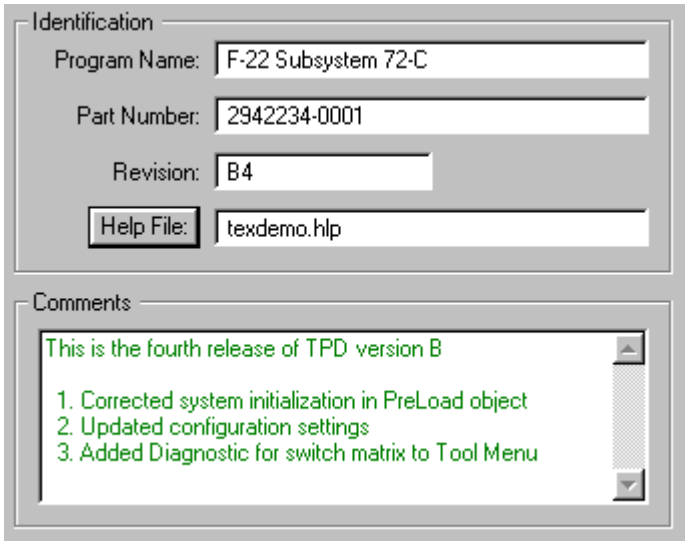
Note: Some error and warning messages respond to a double-click of the left mouse button, or cursor key scrolling. This causes the source of the error, or warning, to be displayed. This is primarily used to highlight problems detected during file load or validation.

All of the information displayed in this window is also written to a log file (TpdBuilder.log). Once this file reaches approximately 200K, it is copied to another file (TpdBuilder.bak) as a buffer. The buffer file is deleted when the next one is copied over it. The log and backup files are created within the common application data directory (in subdirectory "\SSI Test Executive").

Note: Since system behavior and configurations vary widely, you may need to perform a file search in order to determine the exact location of these files. Be sure to include hidden folders in the search.

2.6 Identification

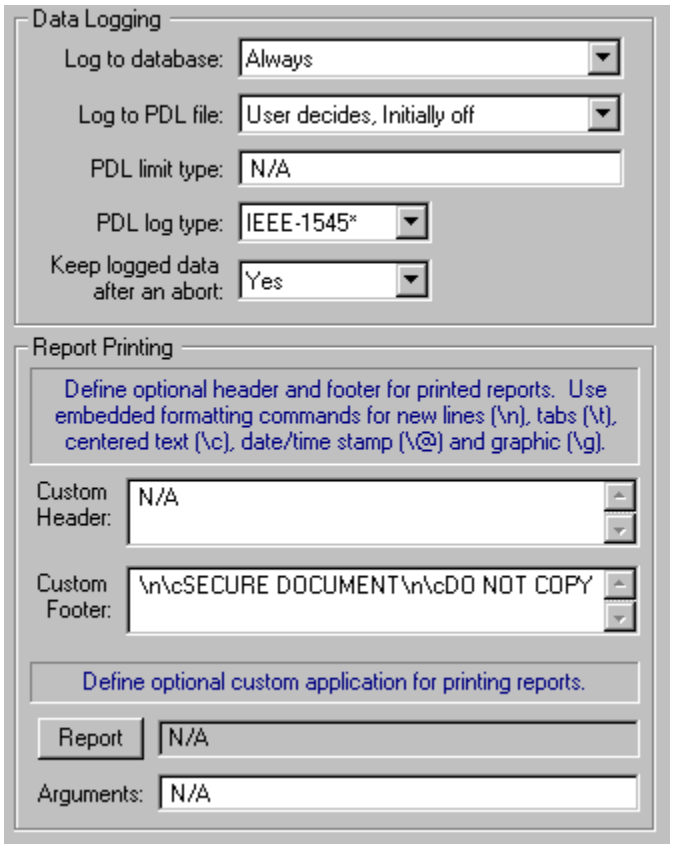
The TPD Builder Identification window is where the TPD name, part number and revision are entered. These must be defined in order for a TPD to be loaded by the Test Executive. The help file entry is optional, pressing the **Help File** button activates a dialog for locating the file. If TPD documentation is assembled into a Windows help file, this is where it is specified. This help file is then accessible from the Test Executive's Help Menu or by pressing the **F1** function key. The help file must reside in the same directory as the TPD file. Separate help files can also be associated with individual test objects in the Test Objects window. The comments entry is available for versioning information or other notes.



2.7 Options

2.7.1 Logging & Printing

The TPD Builder Logging & Printing window is where data logging and report printing options are defined. The logging section controls how and where test results are recorded. This includes how much, if any, the operator controls the logging behavior. The report printing section defines an optional header and/or footer for printed reports. A custom application for generating printed reports can also be specified in this section.



In the data logging section, dropdown lists are used to select logging options. Each list has a **Use Default** entry which defers to the Test Executive's default behavior. A trailing asterisk (*) indicates which of the selections correspond to the default behavior.

The **PDL limit type** enables the logging of measurement limits to a PDL file. The specified string is used as the <limit type> for the DEFINE_LIMIT statement in the PDL file (e.g. DEFINE_LIMIT SPECIFICATION = Measure VCC, >4.85, <5.10;). An entry of **N/A** causes this option to be ignored.

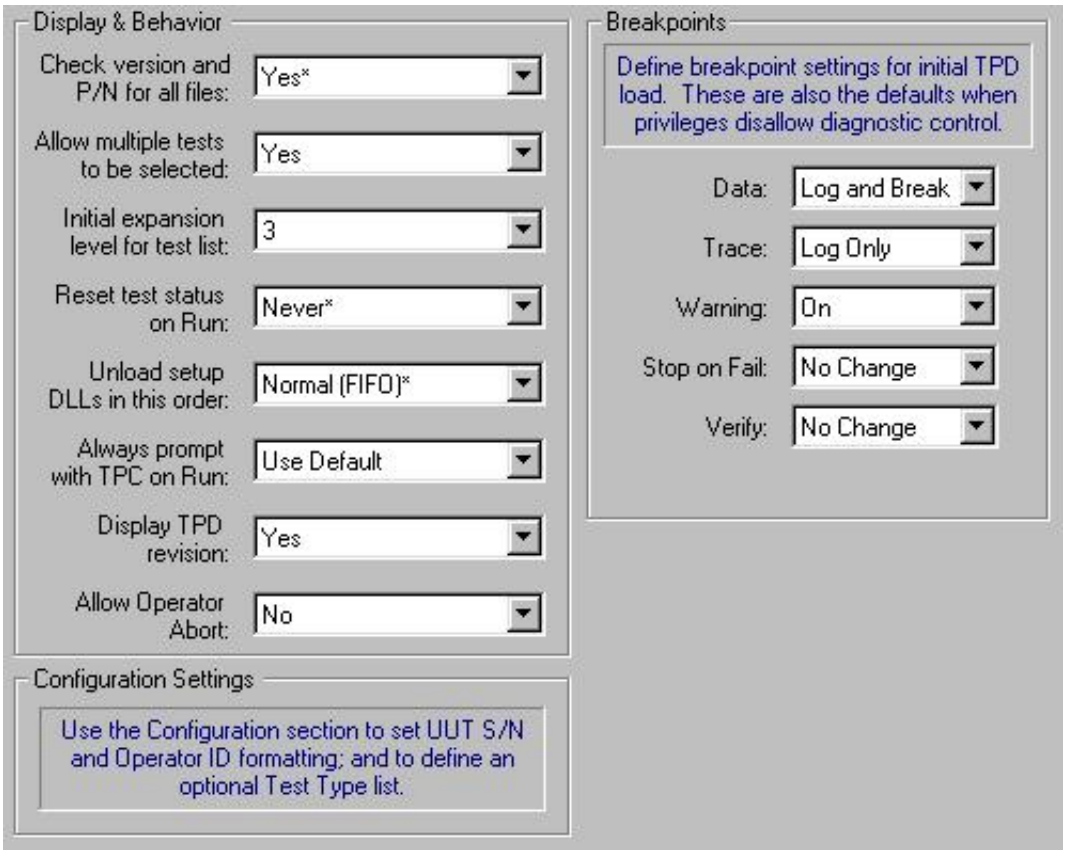
The **Custom Header** and **Custom Footer** entries allow additional information to be added to the printed reports created by the Test Executive and Report Generator. This information might include security warnings, tracking numbers or test constraints. These entries can each be up to 255

characters long. Embedded formatting commands allow new lines (\n), tabs (\t), centered text (\c) and a date/time stamp (\@) for when it was printed. If a header has to be the absolute topmost item on a datasheet, another formatting command (\l) causes it to be placed above the standard printed line. The datasheet header also supports an embedded graphic (\g), which is always printed in the upper right corner of the datasheet. The graphic information is read from a file (**ReportHeader.bmp**) that can reside in the TPD directory, its parent directory or the application directory (typically C:\Program Files\SSI Test Executive). The graphic file is searched for in that order, so different graphics can be associated with different TPDs.

The optional custom report printing application is invoked when “Print Summary Datasheet” is selected from the Test Executive's File Menu. A custom application can be used to meet requirements for printed reports that are beyond the capabilities of the Report Generator. When activated, the application is executed with a command line containing the current UUT database name (enclosed in double quotes if its directory path contains spaces) followed by any items from the **Arguments** entry. This custom application can be built with a variety of tools including Microsoft Access, Visual Basic or Crystal Reports. The internal structure of the UUT database is defined in the Test Executive documentation.

2.7.2 Display & Behavior

The TPD Builder Display & Behavior window is where many operational and presentation options are specified. The left side of the window defines test hierarchy operations, file version checking, setup object DLL unloading, and configuration prompting. Each dropdown list has a **Use Default** entry, which defers to the Test Executive's default behavior. A trailing asterisk (*) indicates which of the selections correspond to the default behavior.



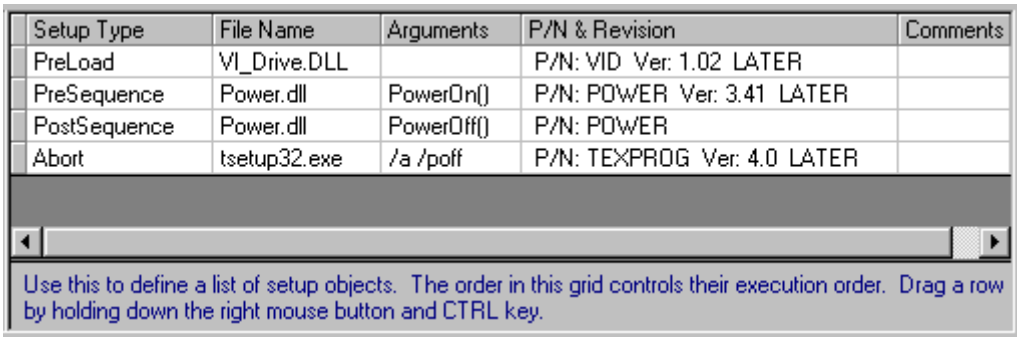
The Breakpoints section defines the initial breakpoint settings for when the TPD is loaded. Creating test procedure documentation is easier when the behavior of the Test Executive, following a TPD load, is predictable. These are also the settings that are reverted to when an operator's privileges disallow diagnostic operation. The **No Change** selection leaves the breakpoint setting to whatever is currently set in the Test Executive.

> **Note: UUT S/N and Operator ID formatting, usually defined within TPD Options, is handled in the Configuration window. This is also where custom Test Types are specified.**

2.8 Setup Objects

The TPD Builder Setup Object window defines executable programs, or DLLs, that are to be run at various times during the testing process. They can be used to initialize instruments, prepare a UUT for power, or release resources following a test sequence. Multiple programs can be associated with each of the setup types. The order that they are entered in this list is the order in which they are executed.

Setup objects can also be DLLs that are loaded for the duration of a test sequence, MSN or TPD. Keeping a DLL loaded facilitates information sharing between test objects and can avoid the reloading of instrument drivers. For more information on using setup objects in a TPD file, see the *Setup Objects* section of the Test Executive manual or help file.



The **Setup Type** column has a dropdown list for identifying the type of setup object. This defines when the setup object is executed during the testing process:

PreLoad	Executed immediately after TPD is loaded
PostLoad	Executed just prior to a TPD being unloaded
PreMSN	Executed when a new UUT S/N is entered
PostMSN	Executed before a new UUT S/N is accepted
PreSequence	Executed at the beginning of a sequence of tests
PostSequence	Executed at the end of a sequence of tests
AbortTest	Executed when an abort occurs, after the test object halts

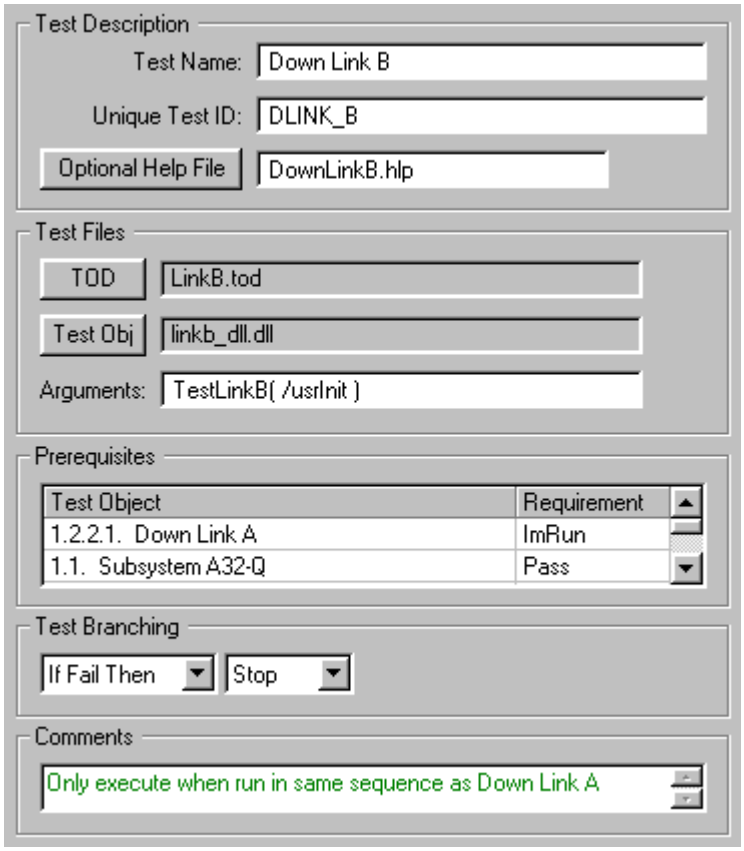
Double click the **File Name** field, to activate a File Definition dialog, to select a file and define its version settings. The **Arguments** field is available to define command line arguments to pass to the setup object when it is executed. To execute a setup object DLL, the arguments field specifies the DLL function to call. Command flags are then specified in trailing parenthesis: PrePowerTest(**/usrGnd J4-16**). Different functions can be called within the same setup object DLL. An execution DLL is always loaded, executed and unloaded. To keep it loaded during testing, preload it prior to execution.

Use the **Comments** field to enter any additional information about the setup object such as its author, its function within the test environment, dependencies with other modules, etc. Use the Edit Menu, or right-click menu, to insert (**Ctrl+Ins**) or Delete (**Shift+Del**) entries in the grid. Rows can be dragged up or down the list by holding down the right mouse button and the **Ctrl** key.

Note: If Vista's "application setup detection" is active, setup objects cannot be executed if they appear to be setup or installation programs. This is because Vista examines any program being started and heuristically detects setup and installation programs based on a variety of factors including file name and internal resource properties.

2.9 Test Objects

The TPD Builder Test Object window defines all of the properties for a single test object or test group. This window displays the properties associated with the test object selected on the hierarchy list. The sections in this window include Test Description, Test Files, Prerequisites, Test Branching and Comments. Note that test groups do not have a Test Files section because they do not support an executable or TOD file.



The Test Description section contains the test name, optional test id and optional help file. The test name is how the test is identified in the Test Executive's Test List window. The optional **Unique Test ID** allows links to a help file for context sensitive support, and is also useful for tracking prerequisites and test branching. This controlled reference name is also used for commands and queries from external applications such as client agents.

The **Optional Help File** entry allows a test object to reference its own help file rather than the common help file defined in the Identification window. Help files for individual test objects must reside in the same directory as the test object executable.

The Comments section allows remarks to be associated with an individual test object or test group. This can be used to document changes or to describe the behavior of the test entry.

2.9.1 Test Files

The Test Files section, of the Test Objects window, defines the test object executable, program arguments and optional TOD file. When a TOD file is specified, the test object can verify measurements against limits stored externally in the TOD file. This provides the flexibility to track test limits separately from test code. Thus test limits can be updated without having to recompile test executables. See the TOD Builder for more information on creating TOD files.

The test object file can be a standalone executable (EXE), a dynamic link library (DLL) or a Visual Basic ActiveX component (DLL). It is defined by pressing the **Test Obj** button which activates a File Definition dialog. The **Arguments** field is where command line arguments are defined for a test object. These command flags, or switches, can be useful for passing information to the test object such as configuration (/usrRAM 0xE000) or test options (/usrPassFailOnly). In order to avoid a conflict between TEXDLL command flags and those used for a test object, it is strongly recommended that test object command flags be prefaced by "**/usr**" (e.g. /usrTest21). The Test Executive is guaranteed never to use command flags prefaced in this manner. This field also provides an opportunity to override breakpoint command flags generated by the Test Executive. For example, if you want to ensure that verify breakpoints are never enabled for a test object, add **/vn** to its argument field. This then has precedence over any verify breakpoint commands defined by the Test Executive.

When a test object is a DLL, the arguments field is also used to specify the DLL function to execute. Command flags are then specified in trailing parenthesis: FilterTest(**/usrFreq 1000**). Different functions can be called within the same test object DLL. A test object DLL is always loaded, executed and unloaded. To keep it loaded during testing, preload it as a setup object DLL.

> **Note: Holding the mouse cursor over either test file causes a tooltip to be displayed containing the file version information. The tooltip is suppressed if version information is undefined or disabled.**

> **Note: If Vista's "application setup detection" is active, test objects cannot be executed if they appear to be setup or installation programs. This is because Vista examines any program being started and heuristically detects setup and installation programs based on a variety of factors including file name and internal resource properties.**

2.9.2 Prerequisites

The Prerequisite section, of the Test Objects window, contains a grid for defining up to 20 optional prerequisite conditions for a test object or test group. Prerequisites are used to control the execution order of test objects. This allows test objects to assume certain initial conditions. For example, a power supply calibration test might be set as a prerequisite for any test object that applies power to the UUT. An unfulfilled prerequisite causes the Test Executive to hold (not run) the dependent test object. The following are some key points concerning prerequisites:

- Prerequisite types are Pass, Run, Immediate Pass and Immediate Run.
- A child or subordinate test inherits prerequisites from its parents.
- For a long list of test objects that must be run in order, make each a prerequisite of the next (i.e. like a linked list).
- Changes to the test object hierarchy can cause difficulties when locating prerequisite test objects. For more information, see the following section, *Resolving Test Object Links*.

The **Requirement** column in the grid selects from the following prerequisite conditions:

Type	Description
Run	Specified test must have been executed.
ImRun	Specified test must be executed in the same sequence.
Pass	Specified test must have been executed and passed.
ImPass	Specified test must be executed in the same sequence and pass.

The two "immediate" prerequisites are used to enforce a specific sequence of test execution. If Test B expects the UUT to be in a particular state following Test A, it can use an Immediate Pass prerequisite to guarantee that A is executed in the same test sequence.

> **Note: Prerequisites make tests conditional on other items in the test list. These conditions are not visible. Consequently, prerequisites should be used sparingly and they should be prototyped with a TPD file before interdependent test objects are created.**

2.9.3 Test Branching

The Test Branching section, of the Test Objects window, has a variable row of dropdown boxes for defining an optional test branch operation. The initial dropdown box selects between the basic test branch commands (**Stop**, **Goto**, **Repeat**). Once a command is selected, a conditional dropdown box appears to its left. This allows the command to be conditional on the test object's, or test group's, execution status (i.e. pass or fail).

The **Repeat** command allows a test object, or test group, to be repeatedly executed a defined number of times. A secondary conditional is available to control the repetition cycle. For example, the repetition might continue as long as there is a passing execution result:

Alternately, the repetition might continue as long as there are less than two execution failures:

Finally, test execution can move elsewhere in the test hierarchy by using the **Goto** command. This can also be conditional on the test object's, or test group's, execution status:

> **Note: Changes to the test object hierarchy can cause difficulties when locating test branch destinations. For more information, see the following section, *Resolving Test Object Links*.**

For more information on using test branching in a TPD file, see the *Test Branching* section of the Test Executive manual or help file.

2.9.4 Resolving Test Object Links

When changes are made to the test object portion of the TPD Hierarchy List, corresponding links for test branching and prerequisites must change also. Whenever possible, the TPD Builder automatically handles these changes.

When a test object is selected, its prerequisites and branch targets are verified and/or recomputed. If a change is detected, a warning or error is displayed in the Debug Log window. Warnings are issued when a target has moved from its original position. Errors are issued when a target cannot be located. These warnings and errors are displayed whenever the test entry is selected, until the test entry is edited. Once modified, the new locations for prerequisites and test branches are stored with the test entry.

To avoid linkage problems due to hierarchy changes, use unique test IDs for test entries. This lets each test be individually identified, and makes it independent from potential rewording of the test name. Even better, also use unique test names for all test entries. Duplicate names and IDs are noted when encountered during automatic link resolution.

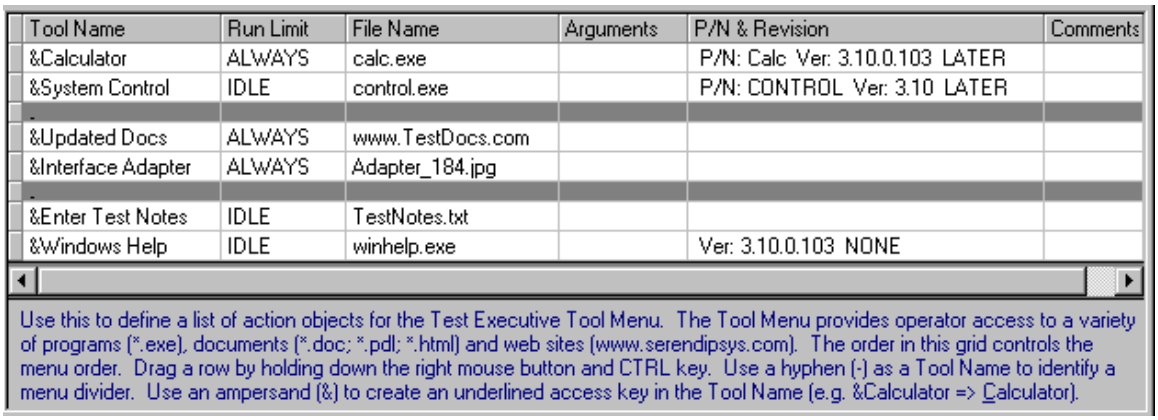
> **Note: When working with complex hierarchies and lots of indentation, enable paragraph numbering, on the test hierarchy, to better target selections for prerequisites and test branches.**

The algorithm, for resolving test object links, tries to match as many elements as possible in order to locate the new position of the target test entry. It first starts with the original position in the test list and looks for a match of paragraph number, test ID and/or test name. To confirm the link, two of the items must match the expected values (unless name and ID are missing). If the target's position in the test list has changed, then a search for a matching paragraph number is performed. If still no confirming name or ID matches, a search with just the name and ID is implemented. If none of these steps results in a matching test entry, an error message is issued for an unresolved test reference.

2.10 Action Objects

The TPD Builder Action Objects window defines a list of programs and documents that are to be placed on the Test Executive's Tool Menu. These programs are made available to the test operator in order to assist in the testing process. They might include diagnostic applications, instrument soft front panels, auxiliary documentation or an Internet/intranet web site.

The **Tool Name** field defines the label that will appear in the Test Executive Tool Menu. Use a preceding ampersand (**&**) to identify a letter as an access key for keyboard control (e.g. Calculator, System Control). Using a hyphen (**-**) for a label creates a dividing line in the Tool Menu (and a dark gray row in the grid).



Tool Name	Run Limit	File Name	Arguments	P/N & Revision	Comments
&Calculator	ALWAYS	calc.exe		P/N: Calc Ver: 3.10.0.103 LATER	
&System Control	IDLE	control.exe		P/N: CONTROL Ver: 3.10 LATER	
-					
&Updated Docs	ALWAYS	www.TestDocs.com			
&Interface Adapter	ALWAYS	Adapter_184.jpg			
-					
&Enter Test Notes	IDLE	TestNotes.txt			
&Windows Help	IDLE	winhelp.exe		Ver: 3.10.0.103 NONE	

Use this to define a list of action objects for the Test Executive Tool Menu. The Tool Menu provides operator access to a variety of programs (*.exe), documents (*.doc; *.pdl; *.html) and web sites (www.serendipsys.com). The order in this grid controls the menu order. Drag a row by holding down the right mouse button and CTRL key. Use a hyphen (-) as a Tool Name to identify a menu divider. Use an ampersand (&) to create an underlined access key in the Tool Name (e.g. &Calculator => Calculator).

The **Run Limit** field selects between ALWAYS and IDLE. These indicate when the tool is allowed to be run. IDLE limits the tool to only run when no testing is active. ALWAYS allows a tool to be run at anytime.

Double click the **File Name** field, to activate a File Definition dialog, to select a file and define its version settings. Besides executable programs, these files can be documents, graphics and some web sites (i.e. those starting with "www."). These are viewed automatically when the Test Executive issues an "Open" command to the application associated with their file type (e.g. *.txt, *.pdf, *.htm, *.jpg). This avoids many of the cumbersome directory paths previously required in order to have an application display a document or picture.

> **Note: A document or graphic file must have an application associated with its file type in order to be displayed properly (e.g. Adobe Reader for *.pdf). You may want to add the application executable (e.g. AcroRd32.exe) to the Other Files list so the Test Executive can ensure its presence on the test system.**

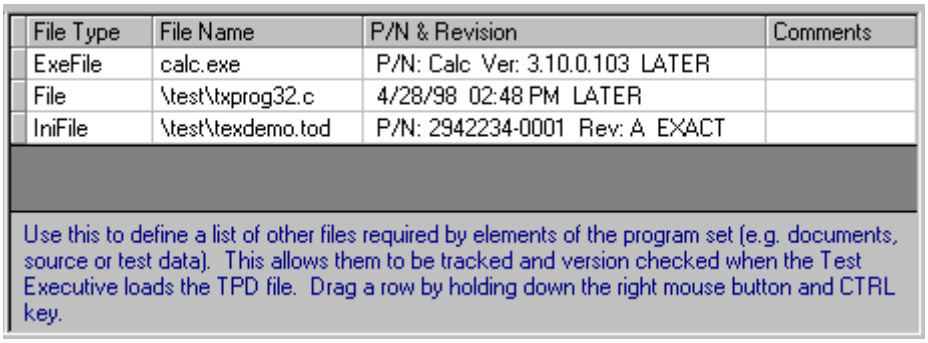
The **Arguments** field is available to define command line arguments to pass to the program when it is executed. Use the **Comments** field to enter any additional information about the program such as its author, its function within the test environment, dependencies with other modules, etc. Use the Edit Menu, or right-click menu, to insert (**Ctrl+Ins**) or Delete (**Shift+Del**) entries in the grid. Rows can be dragged up or down the list by holding down the right mouse button and the **Ctrl** key.

> **Note: If Vista's "application setup detection" is active, items listed in the Tools menu cannot be executed if they appear to be setup or installation programs. This is because Vista examines any program being started and heuristically detects setup and installation programs based on a variety of factors including file name and internal resource properties.**

2.11 Other Files

The TPD Builder Other Files window defines an optional list of files that are necessary for the operation of the test program set. These are files that are not directly used by the Test Executive, but are required by various test objects, setup objects, tools, etc. This file list might include instrument drivers, fault dictionaries, documentation, or other data files.

By listing files in this window, the Test Executive is able to confirm that they are present when the TPD is loaded. Optionally, the part number and version of the files can also be verified. This list additionally serves as a way to document all of the files related to the test program, which can be useful when building an automated installation.


File Type
File Name
P/N & Revision
Comments
ExeFile
calc.exe
P/N: Calc Ver: 3.10.0.103 LATER
File
\test\txprog32.c
4/28/98 02:48 PM LATER
IniFile
\test\texdemo.tod
P/N: 2942234-0001 Rev: A EXACT
Use this to define a list of other files required by elements of the program set (e.g. documents, source or test data). This allows them to be tracked and version checked when the Test Executive loads the TPD file. Drag a row by holding down the right mouse button and CTRL key.

The **File Type** column has a dropdown list for identifying the type of file (*IniFile*, *ExeFile*, *File*). This entry identifies how the version and part number are to be checked for the file. Executable files (e.g. EXE or DLL) have an embedded structure where version and part number (i.e. Internal Name) are stored. INI formatted text files are expected to have an identification section (i.e. [Identification]), with part number and revision keyword entries, similar to those found in TPD and TOD files. All other files are indicated by the generic *File* selection, and version checking is limited to time and date stamps.

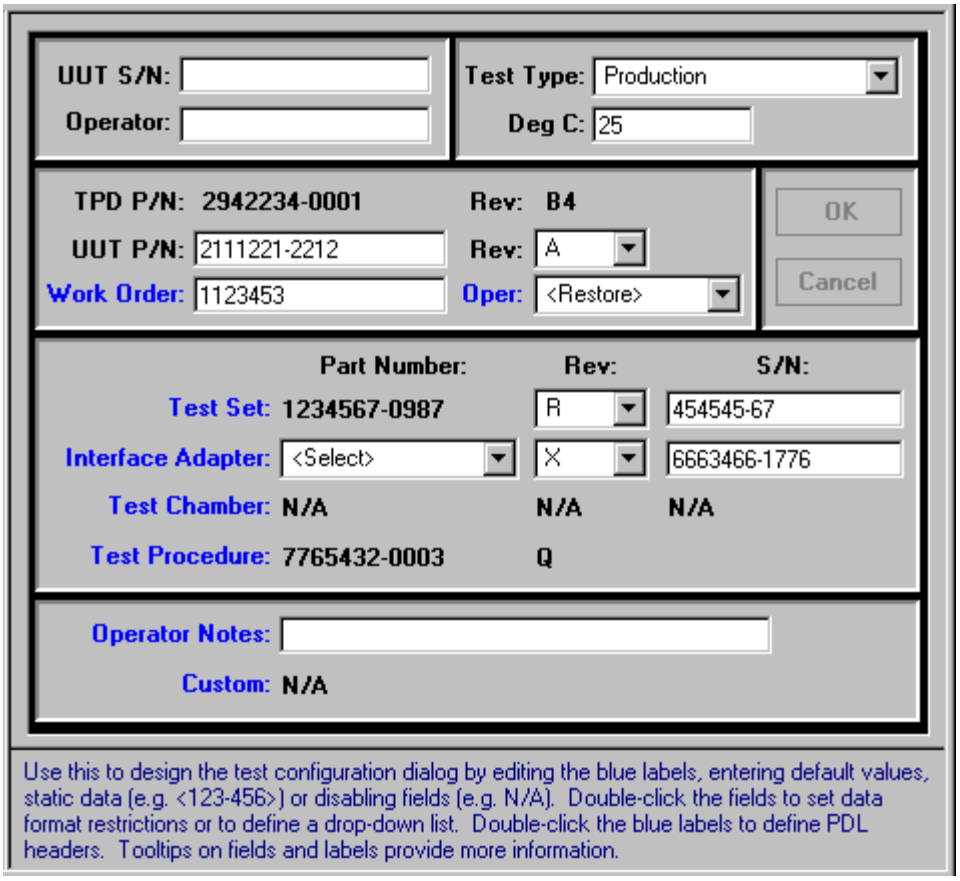
Double click the **File Name** field, to activate a File Definition dialog, to select a file and define its version settings. Use the **Comments** field to enter any additional information about the file such as its author, its purpose within the test set, dependencies with other modules, etc. Use the Edit Menu, or right-click menu, to insert (**Ctrl+Ins**) or Delete (**Shift+Del**) entries in the grid. Rows can be dragged up or down the list by holding down the right mouse button and the **Ctrl** key.

2.12 Configuration

The TPD Builder Configuration window defines the format, labels and defaults for data that are associated with UUT test results. This window contains a duplicate of the Test Executive's Test Configuration dialog. This allows you to see exactly how the dialog will appear when the current settings are applied.

The data entry fields can be directly edited with default values, static values (**<13579-24>**), or disabled (**N/A**). Double-click on a data entry field to active the Configuration Data dialog for entering a format mask, drop-down list or other changes. Left mouse click the down-arrow button on the drop-down controls to see their contents.

The blue labels can also be directly edited in this window. Simply click on one and start typing a new label for the field. Double-click on a blue label, to activate the Custom Label dialog, in order to enter a custom PDL header for the corresponding data fields.

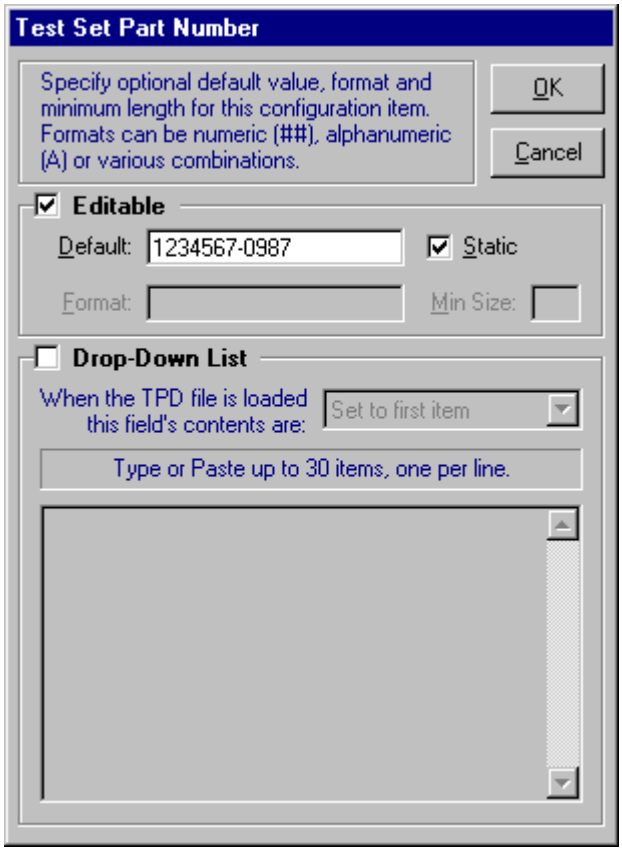


The **UUT S/N** and **Operator** fields only support a format mask for controlling data entry. Double-click their data fields to enter a format mask in the resulting dialog box.

The test type dropdown list has a set of four default types (*Acceptance*, *Manufacturing*, *Calibration*, *Engineering*). Double-click on the drop-down control, or left mouse click the **Test Type** label to activate the Custom Test Types dialog. This allows you to define a custom list of test types. Note that you need to enter the default test types, in your custom list, if you want them included.

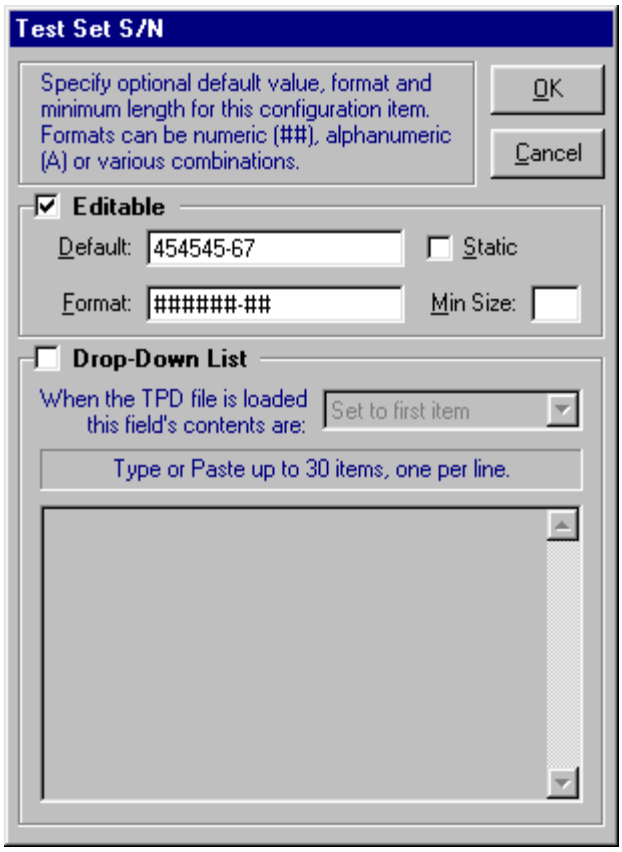
2.12.1 Configuration Data Dialog

The Configuration Data dialog is activated when a data field, in the Configuration window, is double-clicked. This dialog is used to define the contents and behavior of a configuration data entry. The name of the data entry field is shown in the title bar of this dialog box. The **Editable** checkbox controls editing parameters for this configuration item. A **Default** value can be entered in the corresponding text box, and made static (i.e. not changeable by an operator) by clicking the **Static** checkbox. The **Drop-Down List** checkbox activates a section for creating a list of selectable items. Note that only one section checkbox can be active at a time, though they both can be inactive if the data entry field is unused (N/A).


Test Set Part Number
Specify optional default value, format and minimum length for this configuration item. Formats can be numeric (##), alphanumeric (A) or various combinations.
OK
Cancel
Editable
Default: 1234567-0987
Static
Format:
Min Size:
Drop-Down List
When the TPD file is loaded this field's contents are:
Set to first item
Type or Paste up to 30 items, one per line.

Note: Default values, static values (e.g. <AE679-B>) and unused fields (e.g. N/A) can all be achieved by editing directly on the Configuration window.

If the data entry field is not static, an optional format mask can be defined for the data. A format mask controls the characters, length and spacing of a data entry. For example, a UUT's serial number might be a six digit numeric value. By defining a format mask (e.g. ######), only the correct number and type of characters are allowed to be entered. The optional minimum size field (**Min Size**) defines a minimum length for an entry, whereas the format mask sets a maximum length.

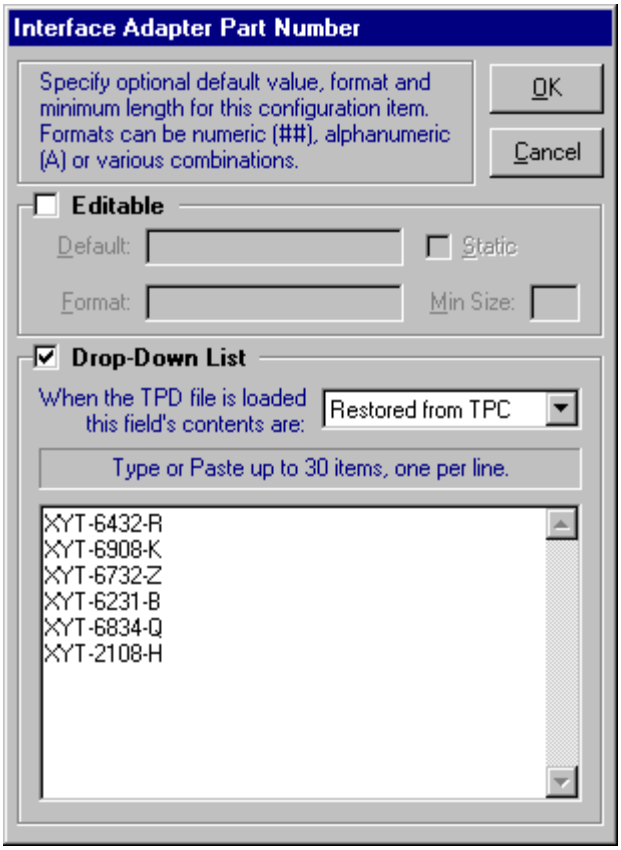

Test Set S/N
Specify optional default value, format and minimum length for this configuration item. Formats can be numeric (##), alphanumeric (A) or various combinations.
OK
Cancel
Editable
Default: 454545-67
Static
Format: ######-##
Min Size:
Drop-Down List
When the TPD file is loaded this field's contents are:
Set to first item
Type or Paste up to 30 items, one per line.

When working with masked edit formats, a default value must contain the same literals as the format mask. For example, if a keyword has a format mask of "##-##", a default value should be entered as "33-33". If a minimum of two characters is acceptable in this instance, the literal must still be entered as "33-". See *Appendix A - Formatting Mask* for more information on formatting masks and literals. Note that when a format mask is defined for a data field, underscores are used as placeholders. Therefore, underscores are not allowed as part of the entered data.

A format mask can include optional placeholders for characters (e.g. "a" = optional alphanumeric character). If optional placeholders are used, it is necessary to also specify a minimum number of characters for the data.

If no minimum size is specified for a keyword, the data entered in the Test Configuration window must exactly match the format specified. If no format is specified, a single character suffices. The format field also controls the maximum size of the data entered.

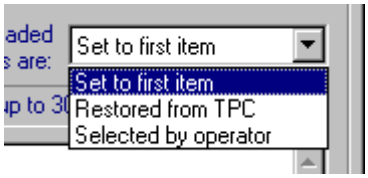
When the **Drop-Down List** checkbox is clicked, the lower portion of the Configuration Data dialog is enabled. This activates a text box for entering a list; and a combo box for selecting an initialization setting.



The contents of the drop-down list are entered in the text box, one per line. A right mouse click activates an editing menu for operations such as Cut, Copy and Paste. Shortcut keys are also supported for these operations (**Ctrl+X**, **Ctrl+C**, **Ctrl+V** respectively). Note that you can cut and paste between this text box and a variety of other applications on your computer.

Up to 30 items can be entered for a drop-down list. Due to TPD formatting restrictions, no items can contain a semicolon (**;**) or comma (**,**).

In the drop-down list section, the combo box selects how the field is initialized when the TPD file is loaded. As a default, the first item in the list is automatically entered in the field when the TPD file is loaded (i.e. **Set to first item**).



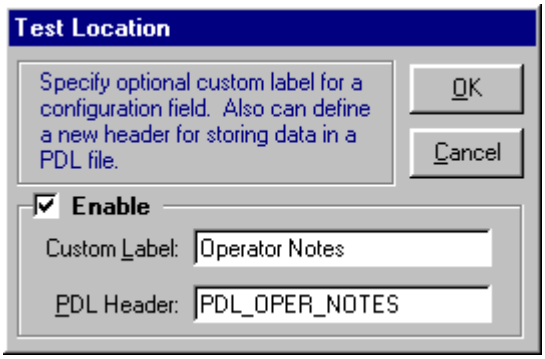
Optionally the field's contents can be restored to their most previous value (i.e. when the TPD was unloaded last) by selecting: **Restored from TPC**. The information to be restored is thus read from the TPC file. If the information in the TPC file does not match the current contents of the drop-down list, which is very common during TPD development, the field is left blank.

In some situations there is a requirement that the test operator actively enter certain information before testing begins. Under those circumstances, the combo box should be set to: **Selected by operator**. When this option is active, the test operator is required to choose an item from the list before the Test Executive runs its first test (following a TPD load). Subsequent tests will run with whatever the operator has picked for that field. So, this option prevents a field's contents from being set to a default value, or restored from the TPC file, when the TPD file is reloaded. Fields with the **Selected by operator** option are initially displayed as blank.

For the purposes of design visibility, these initialization options are displayed on the TPD Builder's Configuration window as first item, <Restore> and <Select>.

2.12.2 Custom Label Dialog

The Custom Label dialog is activated when a blue label, in the Configuration window, is double-clicked. This dialog is used to define an optional custom label, and/or PDL header, for a data entry field. Custom labels are supported by the Test Executive in its Test Configuration dialog, and corresponding printed reports.



Note: Custom labels can be edited directly on the Configuration window by simply clicking on them.

For more information on custom labels, refer to the *Configuration Labels* section of the Test Executive manual or help file.

2.12.3 Custom Test Types Dialog

The Custom Test Types dialog is activated when the **Test Type** field, in the Configuration window, is double-clicked. This dialog is used to define a custom list of test types for operations in the Test Executive. There is a set of four default types (*Acceptance*, *Manufacturing*, *Calibration*, *Engineering*) if this option is disabled.

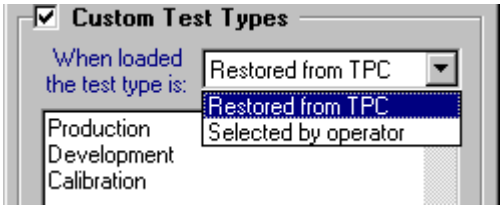
All logged test data is associated with the selected test type. This allows test results to be filtered based on test type. For example, data associated with Calibration or Maintenance procedures might be excluded when processing production test results.

Custom Test Types
Specify optional custom list of test types for test configuration. Enter or Paste up to 20 items, one per line.
OK
Cancel
Custom Test Types
When loaded the test type is:
Restored from TPC
Production
Development
Calibration

The contents of the drop-down list are entered in the text box, one per line. A right mouse click activates an editing menu for operations such as Cut, Copy and Paste. Shortcut keys are also supported for these operations (**Ctrl+X**, **Ctrl+C**, **Ctrl+V** respectively). You can cut and paste between this text box and a variety of other applications on your computer.

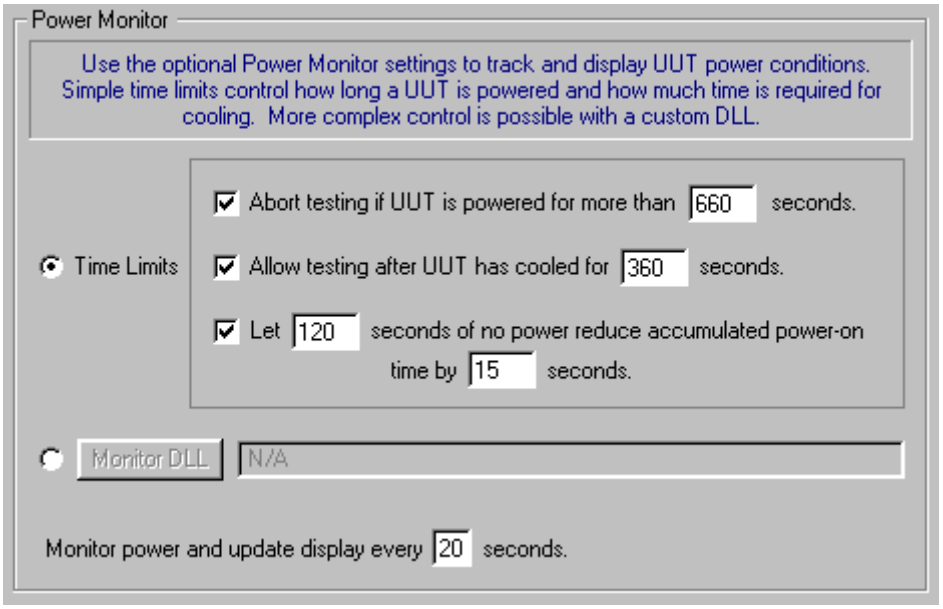
Up to 20 custom test types can be entered in this dialog. Due to TPD formatting restrictions, no test type can contain a semicolon (**;**). Note that you need to enter the default test types, in your custom list, if you want them included.

There are two options for initializing the test type field when the TPD file is loaded. The original behavior is to set the field to its previous value, as read from the TPC file (i.e. **Restored from TPC**). A new option, **Selected by operator**, leaves the field blank when the TPD file is loaded. The blank field thus causes the operator to actively select the test type before testing begins.



2.13 Power Monitor

The TPD Builder Power Monitor window defines optional parameters for tracking the power-on time for a UUT. Since some UUTs have limited cooling capability, this tracking provides a way to abort testing if the power-on limit is exceeded. In addition, a cooling period is supported that disables testing until the UUT has cooled for a specified amount of time. A cooling rate is also available that defines how power-on time is affected when a UUT is idle (i.e. unpowered).


Power Monitor
Use the optional Power Monitor settings to track and display UUT power conditions. Simple time limits control how long a UUT is powered and how much time is required for cooling. More complex control is possible with a custom DLL.
Time Limits
Abort testing if UUT is powered for more than 660 seconds.
Allow testing after UUT has cooled for 360 seconds.
Let 120 seconds of no power reduce accumulated power-on time by 15 seconds.
Monitor DLL
N/A
Monitor power and update display every 20 seconds.

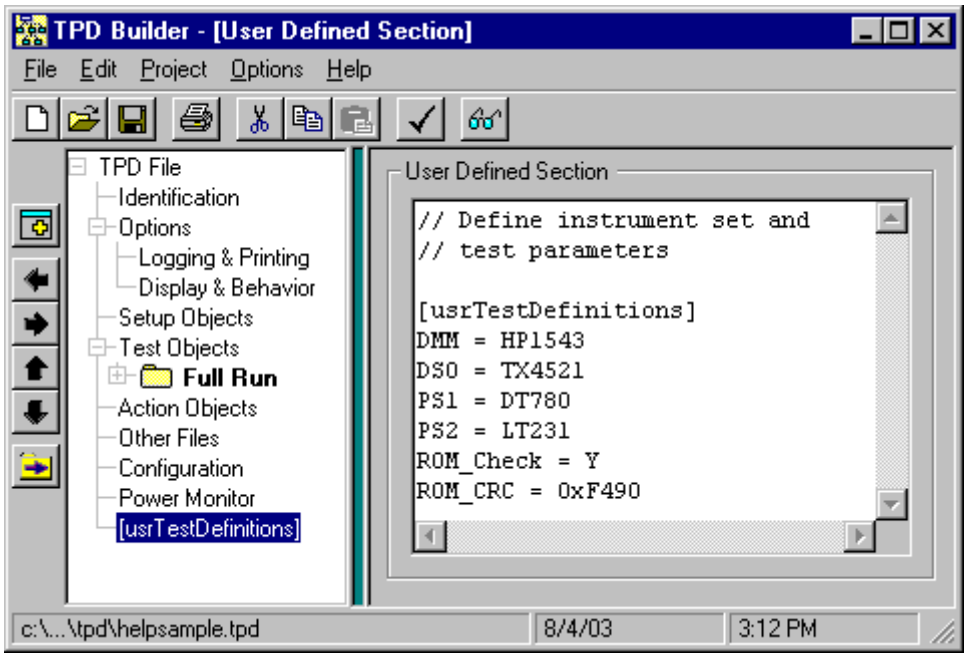
Note: The Test Executive does <u>not</u> control UUT power, it merely tracks the power state defined by test objects via the texPowerMonitor function. When the power-on time is exceeded, one of the abort setup objects <u>must</u> remove power from the UUT <u>and</u> notify the Test Executive with a texPowerMonitor call.

If the built-in algorithm for power monitoring is insufficient for a UUT's requirements, a user-created external DLL can be used instead. The external DLL is notified each time the UUT's power state changes. The DLL then controls the power limits and recovery based on the specific requirements of the UUT.

For more information on using an external DLL for power monitoring, see the *Power Monitor DLL* section of the Test Executive manual or help file.

2.14 User Defined

The TPD Builder User Defined window allows the editing of custom sections of a TPD file. Custom sections are identified by a prefix of "**usr**" in the section name (e.g. [usrTestOptions]). When these are encountered, they are added to the bottom of the TPD Hierarchy List. Selecting a user defined entry causes its contents to be displayed in the text editor window to the immediate right. Standard text editing commands are available from the Edit Menu, right-click menu and toolbar.

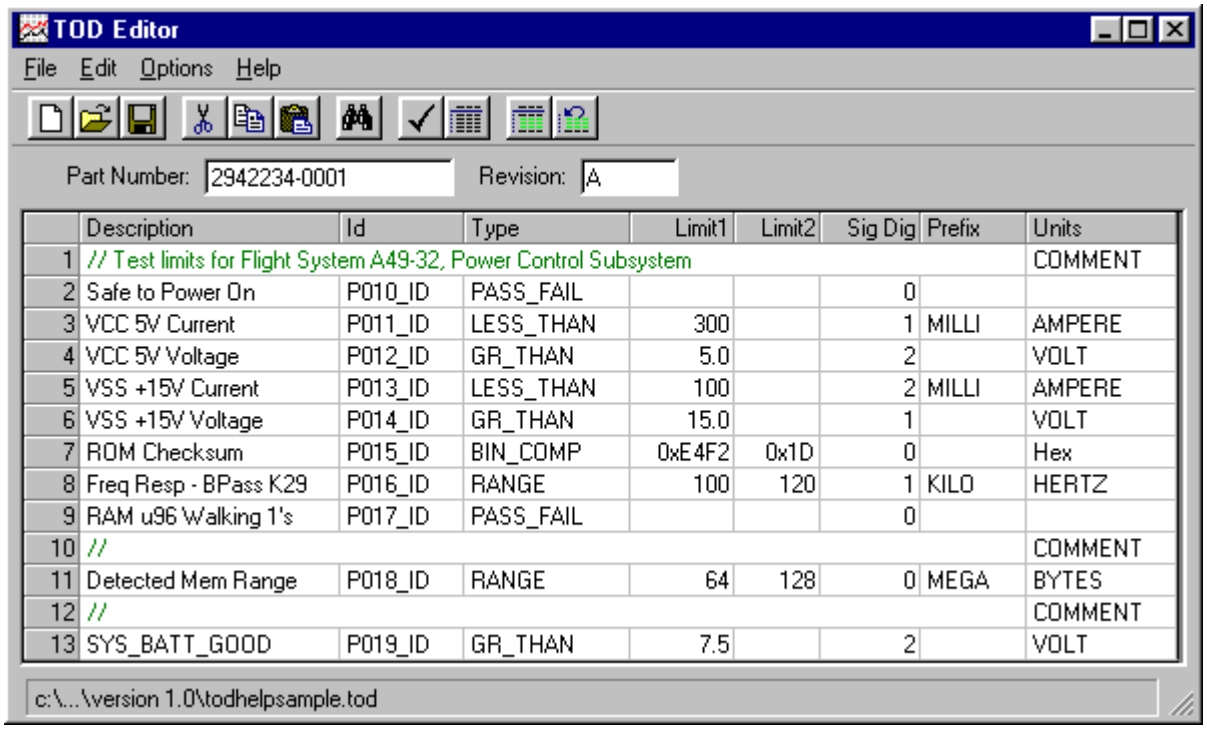


Note: User defined sections must be created by manually adding a header (e.g. [usrNew]) to the TPD file. The TPD Builder does not yet support adding or deleting user defined sections. Once added, the contents of the section can be edited within the TPD Builder.

For more information on user defined items in a TPD file, see the *User Defined* section of the Test Executive manual or help file.

3. TOD Builder

The TOD Builder has an easy-to-use interface for creating and editing a TOD file. With it, a test developer can load a TOD file, edit verify parameters, copy and paste items, validate settings, incrementally fill columns and save in TOD or XML formats.



	Description	Id	Type	Limit1	Limit2	Sig Dig	Prefix	Units
1	// Test limits for Flight System A49-32, Power Control Subsystem							COMMENT
2	Safe to Power On	P010_ID	PASS_FAIL			0		
3	VCC 5V Current	P011_ID	LESS_THAN	300		1	MILLI	AMPERE
4	VCC 5V Voltage	P012_ID	GR_THAN	5.0		2		VOLT
5	VSS +15V Current	P013_ID	LESS_THAN	100		2	MILLI	AMPERE
6	VSS +15V Voltage	P014_ID	GR_THAN	15.0		1		VOLT
7	ROM Checksum	P015_ID	BIN_COMP	0xE4F2	0x1D	0		Hex
8	Freq Resp - BPass K29	P016_ID	RANGE	100	120	1	KILO	HERTZ
9	RAM u96 Walking 1's	P017_ID	PASS_FAIL			0		
10	//							COMMENT
11	Detected Mem Range	P018_ID	RANGE	64	128	0	MEGA	BYTES
12	//							COMMENT
13	SYS_BATT_GOOD	P019_ID	GR_THAN	7.5		2		VOLT

TOD loading and saving are controlled from the File Menu or toolbar. The test limits are contained in the Verify Grid window where they can be edited. Standard editing commands on the Edit Menu include cut, copy, paste, insert, delete and find. Additional commands available via the Options Menu include validate, column fill and comment block. The Debug Log window is provided to view errors and warnings resulting from validation. The Help Menu accesses online help and version information.

TOD Builder v1.05 supports and validates up to 10,000 verify entries in a TOD file. TOD files this large are supported by SSI Test Executive v6.40 (TEXDLL32.DLL v3.40) and later. **Note that test objects are still restricted to only executing 1000 verifies.**

3.1 Verify Grid

The Verify Grid is the primary data entry interface for the TOD Builder. The grid is styled after an electronic spreadsheet with predefined columns and a user-defined number of rows (i.e. verifies). One or more selected cells are highlighted by a yellow background. Comments are shown in green text and content errors are red/bold text (see figure below). To begin editing a selected cell, press the **Enter** key, double click with the mouse or simply start typing. You can move between cells with the **Tab** and **Shift-Tab** keys.

	Description	Id	Type	Limit1	Limit2	Sig Dig	Prefix	Units
1	// Test limits for Flight System A49-32, Power Control Subsystem							COMMENT
2	Safe to Power On	P010_ID	PASS_FAIL			0		
3	VCC 5V Current	P011_ID	LESS_THAN	300		1	MILLI	AMPERE
4	VCC 5V Voltage	P012_ID	GR_THAN	5.0		2		VOLT
5	VSS +15V Current	P013_ID	LESS_THAN	100		2	MILLI	AMPERE
6	VSS +15V Voltage	P014_ID	GR_THAN	15.0		1		VOLT
7	//							COMMENT
8	ROM Checksum	P015_ID	BIN_COMP	0xE4F2	0x1D	0		Hex
9	Freq Resp - BPass K29	P016_ID	RANGE	100	120	1	KILO	HERTZ
10	RAM u96 Walking 1's	P017_ID	PASS_FAIL			0		
11	//							COMMENT
12	// The following entry shows a range limit error							COMMENT
13	Detected Mem Range	P018_ID	RANGE	128	64	0	MEGA	BYTES
14	//							COMMENT
15	SYS_BATT_GOOD	P019_ID	GR_THAN	7.5		2		VOLT

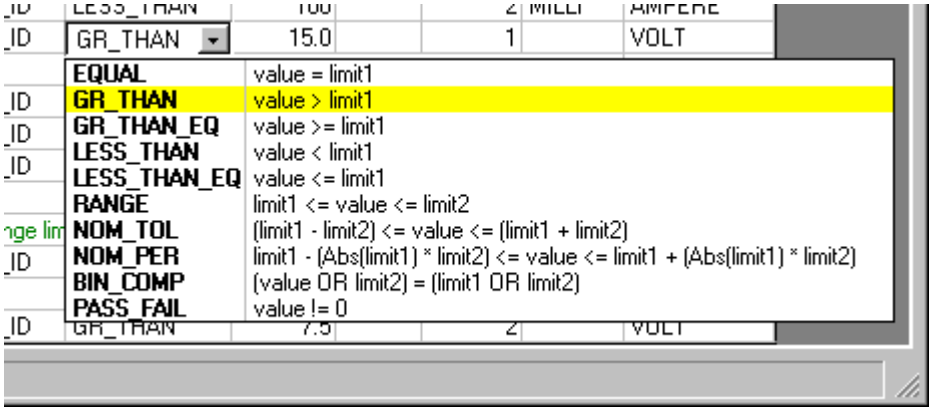
The verify type, significant digits, prefix and units columns all use dropdown lists for selecting appropriate parameters. The Edit Menu has various commands for cutting, copying and pasting both rows and columns. In addition it has commands for row insert, row delete and search. Once a cell is edited, it is immediately checked against the other verify parameters in the row. If an invalid or questionable condition is detected, the cell contents are identified with red/bold text. For a more through evaluation, use the Validate command on the Options Menu. The Fill Column dialog offers a powerful mechanism for initializing all or part of a column with a defined data pattern. These patterns can include an incrementing or decrementing numeric component.

Individual cells are selected by a left mouse click. Multiple cells are selected by holding the left mouse button down and dragging. Large selections are quickly achieved by selecting one corner with a left mouse click and the opposing corner with a Shift-left mouse click. A whole column is selected by a left mouse click on its title. Multiple columns are selected by holding the left mouse button down and dragging it along the title row. A single row is selected by a single mouse click in the leftmost column.

> **Note: Certain desktop color "themes" make it difficult to see the lines in the Verify Grid (e.g. Windows Vista default). There is an optional setting in the TOD Builder Configuration file (TodBuilder.cfg) for setting the grid line color (GridColor). Further information is available in the configuration file.**

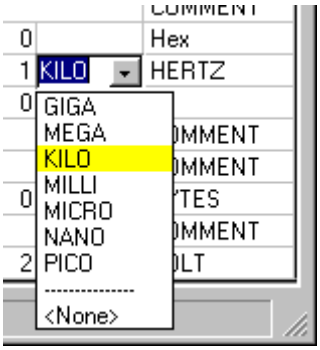
3.1.1 Verify Type Column

The Verify Grid Verify Type column contains a list of comparison types that are available for the verify parameters. This is a fixed list of test methods that the Test Executive supports. By default, the verify types are displayed with all capital letters. Alternate capitalization (e.g. Range instead of RANGE) can be defined in the TOD Builder Configuration file.



3.1.2 Prefix Column

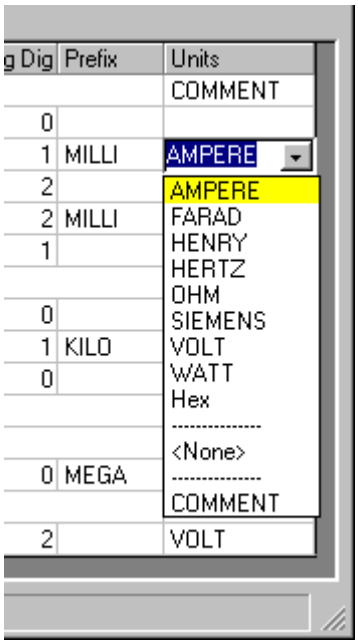
The Verify Grid Prefix column contains a list of prefixes for scaling the measurement units defined in the next column. The default list is targeted for the most commonly used prefixes in engineering. Alternate prefixes can be typed in the textbox or added to a custom list defined in the TOD Builder Configuration file. Prefixes can be up to eight characters. Selecting the **<None>** option indicates no prefix is applicable and the corresponding cell is left blank.



3.1.3 Units Column

The Verify Grid Units column contains a list of measurement units applicable to the associated verify parameters. The default list is targeted for electrical measurements. Alternate units can be typed in the textbox or added to a custom list defined in the TOD Builder Configuration file. Units can be up to 20 characters. The **Hex** option is typically used with a binary compare verify to indicate that the limits and measured value are to be displayed as hexadecimal by the Test Executive. Selecting the **<None>** option indicates no units are applicable and the corresponding cell is left blank.

The **COMMENT** option indicates that the row should be treated as a comment in the TOD file. This allows spacing and documentation to be interspersed with verify parameters. When converting to a comment, verify parameters are encoded in a delimited string that matches a verify line in a TOD file. This allows them to be decoded if the comment is removed later.



3.2 Status Bar

The TOD Builder status bar displays the path and name of the currently loaded TOD file.

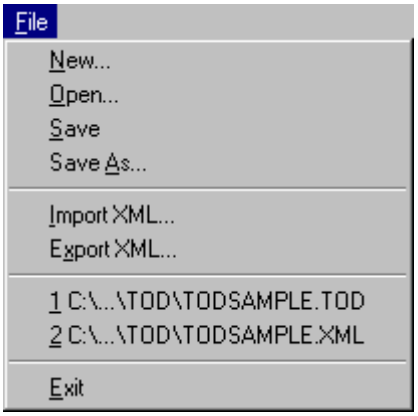
3.3 Toolbar

The toolbar on the TOD Builder provides quick access to commonly used commands. All of the buttons on the toolbar have corresponding menu entries. Many of them also have shortcut keys that perform the same command. The shortcut keys are shown to the right of their menu entries. Tooltips are displayed when the mouse cursor is held over a toolbar button for approximately two seconds. A tooltip box appears temporarily to identify the button's function.

3.4 Menus

3.4.1 File Menu

The TOD Builder File Menu supports loading and saving TOD files. The files can be either in the standard TOD format or XML. When a TOD file is loaded, it is automatically validated and the results are written to the Debug Log window. A file history list permits quick reloading of recently accessed TOD files.

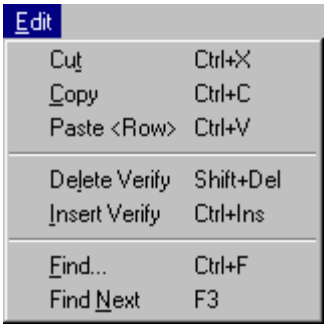


Menu Option	Description
New...	Opens a file browser for naming and placing a new TOD file. A set of default entries are placed in the Verify Grid.
Open...	Opens a file browser for selecting a TOD file. The selected file is loaded and displayed in the Verify Grid.
Save	Updates the TOD file with the latest editing changes.
Save As…	Creates a new TOD file with the latest editing changes. It then becomes the current file.
Import XML…	Opens a file browser for selecting an XML file to import. The imported data overwrites the contents of the TOD Builder.
Export XML…	Creates a specified XML file with the contents of the TOD Builder.
File History	This provides a quick selection from the last four TOD files opened. The list is updated each time a new file is opened.
Exit	Closes the TOD Builder. The current window positions and control selections are saved in an initialization (INI) file.

3.4.2 Edit Menu

The TOD Builder Edit Menu is used to cut, copy and paste portions of the Verify Grid. This can be used to adjust verify positions, duplicate verify entries or to transfer data to another TOD file. The insert and delete commands allow adding or removing individual rows from the grid. To remove multiple rows, use the cut command. When the Find dialog is activated, selected text (blue background) is preloaded for the search

Several toolbar buttons and shortcut keys are provided to more easily initiate these operations.



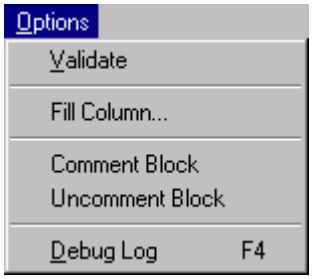
Menu Option	Description
Cut	Copy the selected rows to the system Clipboard, remove them and shift the remaining rows up. When cutting a selection within a column, the selected area is emptied.
Copy	Copy the contents of the selected area to the system Clipboard.
Paste	Insert the contents of the system Clipboard into the grid. Rows are inserted at the topmost selected row. Column data overwrites the selected area.
Delete Verify	Delete the selected verify. The row below is moved up and selected. This allows rapid deletion of multiple rows when used with **Shift+Del**.
Insert Verify	Insert a minimally initialized row above the selected one. Also achieved via **Ctrl+Ins**.
Find…	Activate the Find Dialog to search for a specified item within the Verify Grid. Also activated by **Ctrl+F**.
Find Next	Repeat the most recent search, starting from the currently selected position in the grid. Also achieved via **F3**.

Two types of copy/paste operations are supported. One or more rows can be copied by drag-selecting them down the leftmost column of the grid. Once copied, or cut, the Paste menu entry indicates that **<Row>** data is stored in the Windows clipboard. When pasted, the copied rows are inserted at the topmost currently selected row. Alternately, one or more cells in a column can be selected for copying and pasting to other column cells. When pasting, the currently selected cells are overwritten with the clipboard contents. The Paste menu entry indicates the type of data in the Windows clipboard (**<Row>** or **<Column>**). Column data is simply formatted as a list, one item per line. This allows text to be copied from a variety of sources and pasted into a column in the grid.

Individual cells are selected by a left mouse click. Multiple cells are selected by holding the left mouse button down and dragging. Large selections are quickly achieved by selecting one corner with a left mouse click and the opposing corner with a Shift-left mouse click. A whole column is selected by a left mouse click on its title. Multiple columns are selected by holding the left mouse button down and dragging it along the title row. A single row is selected by a single mouse click in the leftmost column.

3.4.3 Options Menu

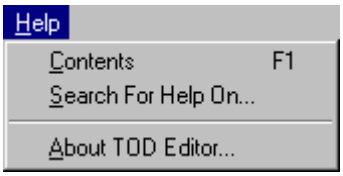
The TOD Builder Options Menu is used to activate special operations for managing the contents of a TOD file. The Validate command reports TOD syntax, length and usage errors in the Debug Log window. Fill Column activates a dialog that fills selected cells with defined values. Comment and Uncomment Block add and remove commented entries in the TOD file.



Menu Option	Description
Validate	Scan TOD contents and report warnings and errors in the Debug Log window.
Fill Column…	Activate the Fill Column dialog to load selected cells with incrementing or decrementing data patterns.
Comment Block	Convert selected rows to comments. Existing verify data is encoded as the comment.
Uncomment Block	Convert selected comment entries to verifies. Encoded verifies from Comment Block are parsed and restored.
Debug Log	Allows the user to toggle between the Debug Log window and the TOD Builder. Function key **F4** also performs this task.

3.4.4 Help Menu

The TOD Builder Help Menu provides access to the online help file and version information. The help file is most easily accessed by pressing the **F1** function key. The F1 key also works for accessing help for dialog windows such as Find and Fill Column.



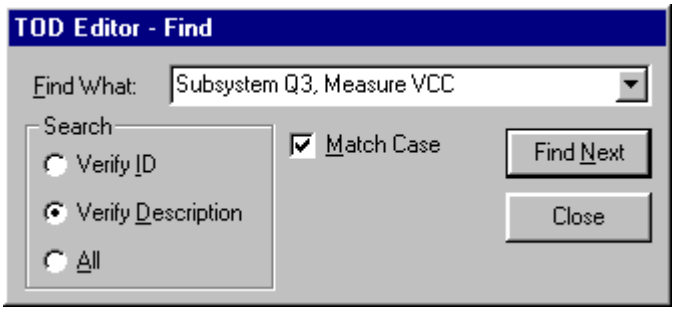
Menu Option	Description
Contents	Displays the Contents page of the help file. Function key **F1** also performs this task.
Search for Help On	Displays the Search index of the help file. Use this to locate specific information in the help file.
About TOD Builder…	Displays a dialog containing version information, a serial number and copyright notice.

Note: Vista users may be prompted to download the Windows Help program (WinHlp32.exe) from the Microsoft support website in order to display files in the Windows Help format (*.hlp).

3.5 Dialogs and Windows

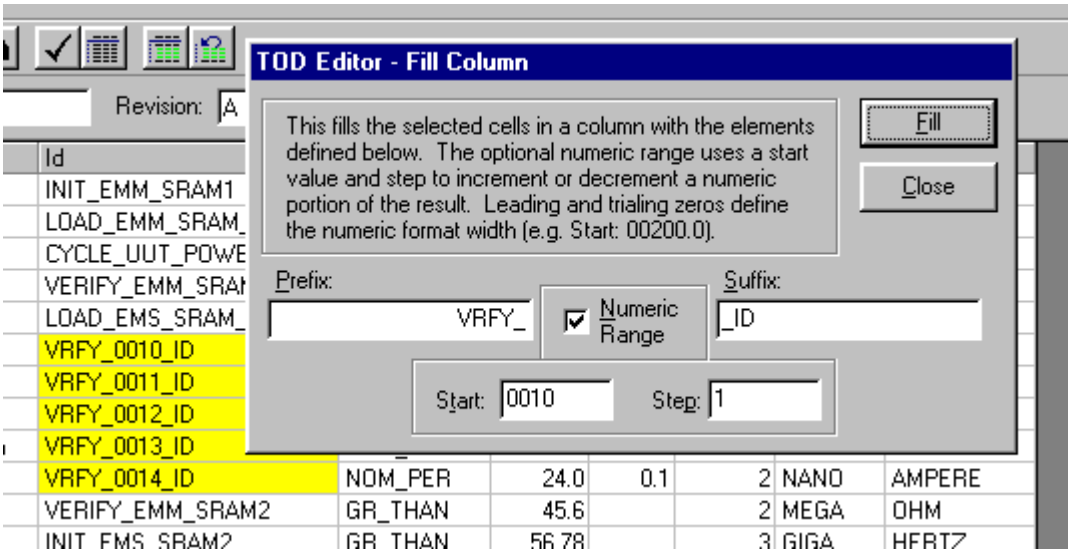
3.5.1 Find Dialog

The TOD Builder Find dialog coordinates a search for the specified text. The search can be restricted to the verify description column, verify ID column or the contents of the whole grid. The search can be made case-sensitive by activating the **Match Case** checkbox. The search text is stored while the TOD Builder is active. This allows previous searches to be accessed via a dropdown list that is part of the text entry. The Find dialog allows multiple searches by remaining active and 'floating' above the TOD Builder window. Use the **Close** button to terminate its operation.



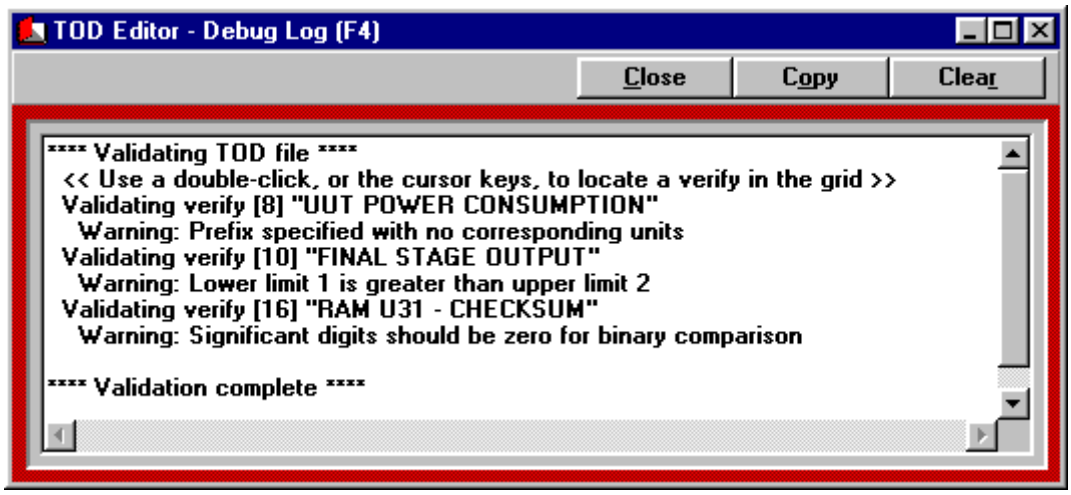
3.5.2 Fill Column Dialog

The TOD Builder Fill Column dialog is a powerful tool for loading columns with defined data patterns. The data pattern is defined as a prefix, suffix and optional numeric portion. When the **Fill** button is pressed, the selected cells are loaded with the specified pattern. The numeric portion of the pattern permits an incrementing, or decrementing, value based on the starting value and the step (i.e. positive or negative). Use leading zeros in the start value to define the width of the numeric portion of the pattern. The Fill Column dialog allows multiple fill operations by remaining active and 'floating' above the TOD Builder window. Use the **Close** button to terminate its operation.



3.5.3 TOD Builder Debug Log

The Debug Log window displays warnings and errors that occur during TOD Builder operations. This includes problems encountered when loading, saving or validating TOD files. The Debug Log window is activated via the Options Menu. Function key F4 toggles between the Debug Log and the TOD Builder window. Scroll bars and key commands (PgUp, PgDn, Home, etc.) are used to move around the window. Push buttons are provided to copy the selected contents to the Windows clipboard or to clear the display.



Note: Some error and warning messages respond to a double-click of the left mouse button, or cursor key scrolling. This causes the source of the error, or warning, to be displayed. This is mostly used to highlight verify entry problems detected during file load or validation.

All of the information displayed in this window is also written to a log file (TodBuilder.log). Once this file reaches approximately 200K, it is copied to another file (TodBuilder.bak) as a buffer. The buffer file is deleted when the next one is copied over it. The log and backup files are created within the common application data directory (in subdirectory "\SSI Test Executive").

Note: Since system behavior and configurations vary widely, you may need to perform a file search in order to determine the exact location of these files. Be sure to include hidden folders in the search.

4. Operational Notes

4.1 Test Criteria

Verify/compare test criteria is defined by a set of eight parameters:

Parameter	Type	Description
description	string	A descriptive, user-friendly string. Up to 40 characters. Empty strings raise a warning.
id	string	A unique string identifier. Up to 40 characters. This string can only contain alphanumeric characters and underscores. Illegal characters or empty strings are reported as errors.
limitType	short	Determines what test method is used. Legal values for this parameter are defined by a dropdown list
limit1	double	The usage depends upon *limitType* (see *Test Methods*).
limit2	double	The usage depends upon *limitType* (see *Test Methods*).
sigDigits	short	Number of significant digits applied to the test criteria parameters. Legal values for this parameter are zero through six.
prefix	string	The order of magnitude for the units of measurement (e.g. “milli”). Up to eight characters. The contents of the dropdown list can be customized in the TOD Builder Configuration file.
units	string	The units of measurement for the compare (e.g. “volts”). A value of “hex” displays the compare using hexadecimal format. Up to 20 characters. The contents of the dropdown list can be customized in the TOD Builder Configuration file.

4.2 Test Methods

Ten different test methods are available for verifies and compares. The following constant values specify how the test criteria are used to evaluate a given test parameter.

Value	Description
EQUAL	Passes only if *testParam* is equal to *limit1*. Significant digits are applied to *testParam* and *limit1* (*limit2* not used).
GR_THAN	Passes only if *testParam* is greater than *limit1*. Significant digits are applied to *testParam* and *limit1* (*limit2* not used).
GR_THAN_EQ	Passes only if *testParam* is greater than or equal to *limit1*. Significant digits are applied to *testParam* and *limit1* (*limit2* not used).
LESS_THAN	Passes only if *testParam* is less than *limit1*. Significant digits are applied to *testParam* and *limit1* (*limit2* not used).
LESS_THAN_EQ	Passes only if *testParam* is less than or equal to *limit1*. Significant digits are applied to *testParam* and *limit1* (*limit2* not used).
RANGE	Passes only if *testParam* is within an inclusive range whose lower and upper bounds are *limit1* and *limit2* , respectively. Significant digits are applied to *testParam* and both bounds.
NOM_TOL	Passes only if *testParam* is within an inclusive range whose lower and upper bounds are based on a nominal tolerance: The lower bound is *limit1* - *limit2*. The upper bound is *limit1* + *limit2*. Significant digits are applied to *testParam* and both bounds.
NOM_PER	Passes only if *testParam* is within an inclusive range whose lower and upper bounds are based on a nominal percentage: The lower bound is *limit1* - (| *limit1* | * *limit2*). The upper bound is *limit1* + (| *limit1* | * *limit2*). Significant digits are applied to *testParam* and both bounds.
BIN_COMP	Passes only if *testParam* logically ORed with *limit2* is equal to *limit1* logically ORed with *limit2* (significant digits are not used).
PASS_FAIL	Passes only if *testParam* is non-zero (*limit1*, *limit2*, and significant digits are not used).

The BIN_COMP test method implements a “don’t care” mask, such that all bits set in the mask parameter (*limit2*) are ignored by the evaluation. Note that you reverse the mask behavior if you logically invert the mask parameter. This way, you can achieve a “select” mask; such that only the bits set in the mask parameter are tested by the evaluation.

5. Builder Files

5.1 TOD Builder Configuration File

The TOD Builder Configuration file (TodBuilder.cfg) is an INI-style text file that contains optional settings for controlling the behavior of the user interface. Some of the options include: font color for comments and errors; custom lists for prefix and units; capitalization override for verify types and a disable option for displaying exported XML.

For the latest information on the capabilities and syntax of the TOD Builder Configuration file, refer to the comments contained in the file itself.

> **Note: The Configuration file is only read when the TOD Builder first starts. Therefore, restart the TOD Builder in order to see the result of changes.**

5.2 XML Export File

The TOD Builder File Menu allows a TOD file to be imported or exported with an XML format. The following is an example of a simple TOD file in XML format.

```
<?xml version="1.0"?>
<tod_file>
  <part_number>2942234-0001</part_number>
  <revision>A</revision>
  <comment>Limits for Flight System A4-32, Power Subsystem</comment>
  <verify>
    <description>Safe to Power On</description>
    <id>P010_ID</id>
    <type>PASS_FAIL</type>
    <limit1></limit1>
    <limit2></limit2>
    <sig_digits>0</sig_digits>
    <prefix></prefix>
    <units></units>
  </verify>
  <verify>
    <description>VCC 5V Current</description>
    <id>P011_ID</id>
    <type>LESS_THAN</type>
    <limit1>300</limit1>
    <limit2></limit2>
    <sig_digits>1</sig_digits>
    <prefix>MILLI</prefix>
    <units>AMPERE</units>
  </verify>
  <verify>
    <description>VCC 5V Voltage</description>
    <id>P012_ID</id>
    <type>GR_THAN</type>
    <limit1>5.0</limit1>
    <limit2></limit2>
    <sig_digits>2</sig_digits>
    <prefix></prefix>
    <units>VOLT</units>
  </verify>
  <verify>
    <description>VSS +15V Current</description>
    <id>P013_ID</id>
    <type>LESS_THAN</type>
    <limit1>100</limit1>
    <limit2></limit2>
    <sig_digits>2</sig_digits>
    <prefix>MILLI</prefix>
    <units>AMPERE</units>
  </verify>
  <verify>
    <description>VSS +15V Voltage</description>
    <id>P014_ID</id>
    <type>GR_THAN</type>
    <limit1>15.0</limit1>
    <limit2></limit2>
    <sig_digits>1</sig_digits>
    <prefix></prefix>
    <units>VOLT</units>
  </verify>
  <comment></comment>
  <verify>
    <description>ROM Checksum</description>
```

```
    <id>P015_ID</id>
    <type>BIN_COMP</type>
    <limit1>0xE4F2</limit1>
    <limit2>0x1D</limit2>
    <sig_digits>0</sig_digits>
    <prefix></prefix>
    <units>Hex</units>
  </verify>
  <verify>
    <description>Freq Resp - BPass K29</description>
    <id>P016_ID</id>
    <type>RANGE</type>
    <limit1>100</limit1>
    <limit2>120</limit2>
    <sig_digits>1</sig_digits>
    <prefix>KILO</prefix>
    <units>HERTZ</units>
  </verify>
  <verify>
    <description>RAM u96 Walking 1&amp;apos;s</description>
    <id>P017_ID</id>
    <type>PASS_FAIL</type>
    <limit1></limit1>
    <limit2></limit2>
    <sig_digits>0</sig_digits>
    <prefix></prefix>
    <units></units>
  </verify>
  <comment></comment>
  <comment>The following entry shows a range limit error</comment>
  <verify>
    <description>Detected Mem Range</description>
    <id>P018_ID</id>
    <type>RANGE</type>
    <limit1>128</limit1>
    <limit2>64</limit2>
    <sig_digits>0</sig_digits>
    <prefix>MEGA</prefix>
    <units>BYTES</units>
  </verify>
  <comment></comment>
  <verify>
    <description>SYS_BATT_GOOD</description>
    <id>P019_ID</id>
    <type>GR_THAN</type>
    <limit1>7.5</limit1>
    <limit2></limit2>
    <sig_digits>2</sig_digits>
    <prefix></prefix>
    <units>VOLT</units>
  </verify>
</tod_file>
```

5.3 Test Object Definition File

A Test Object Definition (TOD) file is a repository for verify parameters. TOD files are used to isolate test limits from the program that performs the measurements. This allows the limits to be adjusted without rebuilding the executable. This is extremely valuable when adapting tests for engineering changes or when using different limits for production versus environmental test. Limits for several different test objects can be stored in one TOD file. This can make tracking, installing and updating simpler.

Test objects access these parameters when they execute **texVerifyTOD** or **texCompareTOD**. These two functions compare the stored limits with the results of a measurement. Each verify parameter set is identified by a unique identifier. A test object can also retrieve verify parameters from a TOD with the **texGetTODVerify** function. This allows verify parameters to be examined, logged or scaled to meet specific measurement or system requirements.

For more information on TOD file formatting, see *Test Object Definition Format*.

5.3.1 Test Object Definition Format

A TOD file contains a list of verify parameters for automatic consumption by a test object. Each TOD file includes two sections: an Identification section and a Verifies section. The Identification section specifies a part number, revision and TOD format version. The Verifies section contains one or more verify definitions. Each verify definition consists of a line that specifies eight test criteria. For more information on these parameters, see *Test Criteria*.

Additional, user-defined sections are permitted in a TOD file if they are placed between the [Identification] and [Verifies] sections. User-defined sections in a TOD file should be prefaced by "**usr**" (e.g. [usrSection1]) and must not contain the following keywords: PartNumber, Revision, FormatVersion or Verify.

A test object consumes TOD verify data based on the unique identifier field. This allows the TOD file to contain verify data in any order. This protects the test developer from subtle problems due to changes in verify sequencing. A TOD file can hold verify data for several different test objects.

A TOD file is associated with a test object in a Test Program Definition (TPD) file. The TOD file name is contained in the test object definition line.

5.3.2 TOD Identification

This section identifies the TOD part number and revision. It also indicates the TOD format version. All lines in the [**Identification**] section are formatted as:

keyword = *string*

Keyword	Description
PartNumber	Required part number for identifying TOD.
Revision	Required revision number of TOD.
FormatVersion	Required TOD format level (**1.0**).

The Part Number and Revision entries are used to identify the TOD file and to validate it against the TOD part number and version declared in a TPD file.

When a TOD file is created or saved by the TOD Builder, the format version is set to 1.1. This provides a way to distinguish between manually created TOD files (FormatVersion = 1.0) and ones created by the TOD Builder. The only real difference between the formats is that a manually created TOD file may have additional comments, user-defined sections or unique spacing within the verify parameter declarations.

Example

```
[Identification]
PartNumber = 54321
Revision = B
FormatVersion = 1.00
```

5.3.3 TOD Verifies

Each entry in this section of a TOD file corresponds to a single verify operation. All of the information necessary for performing and logging a verify is contained in an entry (except for the value of the actual measurement). A test object matches a measurement to its corresponding limits through the unique identifier field.

All lines in the [**Verifies**] section are formatted as:

Verify = *VerifyName; UniqueID; LimitType; limit1; limit2; SigDigits; UnitPrefix; UnitType*

Field	Description
VerifyName	Descriptive name of verify (up to 40 characters).
UniqueID	Unique identifier for verify (only alpha-numeric & underscore accepted).
LimitType	Required, comparison type: Range, Nom_Tol, Nom_Per, Gr_Than, Gr_Than_Eq, Less_Than, Less_Than_Eq, Equal, Bin_Comp, Pass_Fail.
limit1	Numeric first limit value.
limit2	Numeric second limit value.
SigDigits	Number of significant digits for comparison (i.e. to the right of the decimal point). Range from 0 to 6.
UnitPrefix	Prefix to units (e.g. micro). Use NONE or empty field if no prefix is necessary.
UnitType	Measurement units (e.g. volts). Use NONE or empty field if no units are necessary.

Example

```
[Verifies]
Verify = Measure Power Supply; IDA; EQUAL; 1; 0; 0; MILLI; VOLTS
Verify = Check Noise Calibration; IDB; LESS_THAN; 2; 0; 0; MILLI; VOLTS
// Comments are allowed between verify entries
Verify = Adapter Test; IDC; GR_THAN; 0; 0; 0; NONE; NONE
```

5.4 Test Program Definition File

A TPD file dictates much of the behavior of the Test Executive. It defines external factors, such as the hierarchy of tests, and internal behavior, such as data logging. Setup, testing, operator tools and configuration information are all built into the TPD file.

The format of a TPD file is based on Windows' INI files. This is a text format with bracketed sections (e.g. **[Identification]**) and keywords (e.g. **Database = Always**). Taken individually, the syntax of a single line is not too difficult. Collectively, a large TPD file can be intimidating due to number of options and settings that are possible.

The TPD Builder is structured to match the sections contained in a TPD file. Thus a direct link between the user interface and the resulting file format is maintained. Wherever possible, options and selections are designed to be clearly understood. When necessary, a user should refer to the detailed specification in the *Test Program Definition File* section of the Test Executive manual and help file.

> **Note: When the TPD Builder saves a TPD file, an additional keyword is added to its identification section. This is done so the TPD Builder can issue a warning when a hand-edited TPD file is loaded. Hand-edited TPD files may have spacing, or comments, that will be lost when overwritten by the TPD Builder.**

6. Appendix A - Formatting Mask

A format mask for test configuration data consists of placeholders and literal characters. The placeholders define the types of characters that can be entered (e.g. digits, letters or alphanumeric). The literal characters are typically separators (e.g. the hyphens in a phone number). The following table shows the characters available for defining a format mask.

Mask Character	Description
#	Digit placeholder
.	Decimal placeholder
,	Comma or thousands separator
:	Colon or time separator
/	Slash or date separator
****	Treats the next character in the mask as a literal. This allows the inclusion of the "#", "&", "?", ";" and "A" in the mask.
&	Character placeholder for ANSI characters in the following ranges: 32-126 and 128-255.
>	Convert all the characters that follow to uppercase.
<	Convert all the characters that follow to lowercase.
A	Alphanumeric character placeholder (entry required).
a	Alphanumeric character placeholder (entry optional).
9	Digit placeholder (entry optional).
C	Character or space placeholder (entry optional).
?	Letter placeholder
Literal	All other characters or symbols are displayed as themselves.

Note: When a format mask is defined for a data field, underscores are used as placeholders. Therefore, underscores are not allowed as part of the entered data. You must use "\;" to include a semicolon as a literal character.

When defining a TPD configuration data mask that includes optional placeholders (e.g. 9 or a), it is necessary to also specify a minimum number of characters for the data (e.g. TestProcRevision = ; ?a; 1).

Here are some formats for commonly used data:

Phone Number (U.S.) = **(**###**)** ###**-**####
Social Security = ###**-**##**-**####
Date of Birth = ##**/**##**/**##

7. Appendix B - Version Changes

The following are lists of changes that have occurred from previous versions of the TPD and TOD Builders.

Changes from Release 1.04 to Release 1.05 of the TPD/TOD Builders

1. The TPD Builder now supports **OperatorAbort** in the Display & Behavior Options window. This option works with version 6.40, and later, of the SSI Test Executive.

2. The TOD Builder now supports and validates up to 10,000 verify entries in a TOD file. TOD files this large are supported by SSI Test Executive v6.40 (TEXDLL32.DLL v3.40) and later. **Note that test objects are still restricted to only executing 1000 verifies.**

3. The example TOD Builder Configuration file (TodBuilder.cfg) has been updated with the correct section label: **[TOD Builder]**. This file defines various behavior and display options for the TOD Builder.

4. The TOD Builder Configuration file now supports a display option that sets the color of the grid lines (**GridColor**). This is useful when a system's desktop theme makes the default grid line color difficult to see. There is a separate section/setting for both the TPD and TOD Builders.

5. In order to better operate with Windows Vista, temporary files (*.ini, *.log) from the TPD/TOD Builder home directory have been relocated to the system's common application data directory (in subdirectory "\SSI Test Executive"). A separate application note, included with the Test Executive v6.4 installation, further details operation with Windows Vista

Changes from Release 1.03 to Release 1.04 of the TPD Builder

1. The TPD Builder now supports the definition of drop-down lists in the Configuration window. These lists are defined for most data entry fields via the Configuration Data dialog, which also has optional initialization settings: SELECT and RESTORE. The Custom Test Type dialog also supports the SELECT setting for custom test type lists. These options work with version 6.36, and later, of the SSI Test Executive.

2. Due to the significant addition of drop-down lists to the TPD file syntax, the TPD file format version has been changed from 2.0 to 2.1. TPD files with format version 2.1 can only be loaded by Test Executive v6.36 and TPD Builder v1.04 (or greater). The TPD Builder only assigns format version 2.1 to TPD files that use drop-down lists. A warning is issued if a 2.0 file is being updated to version 2.1.

3. This release adds more default file types to the file dialog when browsing for Action Objects. This is to indicate that Action Objects can now include documents (*.txt, *.pdl, *.html), graphics (*.jpg, *.bmp) and video files (*.wmv, *.mpg).

4. Modified the file browsing sequence to work better with Windows XP. The starting folder (directory) when browsing for test objects and TOD files is now always set to the \Test subfolder relative to the TPD file location. The initial folders when browsing for Setup and Action objects are \Setup and \Tools, respectively.

Changes from Release 1.02 to Release 1.03 of the TPD & TOD Builders

1. The TPD Builder now supports DisplayTpdRevision in the Display & Behavior Options window. This option works with version 6.35, and later, of the SSI Test Executive.

2. The Action Objects' File Name column, in the TPD Builder, now allows web page references (e.g. www.serendipsys.com) and document files (*.txt, *.pdf, *.hlp, etc). These are compatible with version 6.35, and later, of the SSI Test Executive.

3. The TOD Builder's Validate operation now issues an error if more than 1000 verifies are present in the TOD file. The SSI Test Executive only allows up to 1000 verifies in each TOD file.

4. Pasting text into a cell of the TOD Builder grid has been improved. The pasted text is now placed in the cell at the current cursor location, and the cell remains in the edit mode. Pasting into a cell is accomplished with Ctrl+V or Shift+Ins or a popup edit menu (right mouse click). Note: Using the Paste command from the regular Edit Menu replaces the cell contents with the contents of the paste buffer.

5. Corrected a problem with the TOD Builder that occurred when selecting rows for cutting. If the rows were selected from a higher line number to a lower line number, the rows beneath the selection would be cut. This has been corrected and rows can now be selected in either direction for cutting.

Changes from Release 1.01 to Release 1.02 of the TPD Builder

1. The TPD Builder now has integrated on-line help. Press **F1** for help with any window or dialog box.

2. Shortcut keys, on the Edit Menu, have been added and changed to match those on the TOD Builder.

3. A warning has been added when a test group is deleted from the TPD Hierarchy List. Since deleting a test group also deletes its children, a warning was deemed to be necessary.

Changes from Release 1.0 to Release 1.01 of the TPD Builder

1. The TPD Builder now correctly displays test branch destinations when the TPD file is initially loaded.

2. Setup, Action and Other file editing has been updated to fix errors that occurred when all entries are deleted. Several other small errors have also been corrected in this area.

3. Under certain conditions, time/date version checking for a file could not be activated. This has been corrected.

4. Static configuration data cannot be blank because the Test Executive does not accept blank data as being defined. The TPD Builder now overrides this condition with a static hyphen (-).

Changes from Beta Release to Release 1.0 of the TPD Builder

1. The TPD Builder now supports a validation command which checks all test object references, verifies the existence of all specified files and confirms their version information.

2. Warnings and errors printed in the Debug Log Window can now be double-clicked to display the source of the warning or error.

3. Paragraph numbers can now be optionally displayed in the test hierarchy.

4. Prerequisite and test branch references are now better at resolving discrepancies due to editing changes in the test hierarchy.

5. Individual comments are now allowed for each test object entry in the hierarchy.

6. Various improvements have been made to the general editing and display of the TPD contents.

7. **Note:** the TPD Builder currently does not have on-line help.

8. Serendipity Systems, Inc.

Serendipity Systems, Inc. was founded in 1984 to provide the ATE and electronics manufacturing community with test products and engineering services. For over 20 years Serendipity Systems has been involved in the development of custom software and hardware solutions for its clients. This has included ATE runtime software, test program translators, database systems, diagnostic program analysis software and handheld test equipment.

The company has produced custom GUI user interfaces, test programming post-processing tools, open architecture instrument integration strategies and fault isolation software in the form of fault dictionary and guided probe processors.

Serendipity Systems' staff is conversant in commonly used programming languages and systems including "C", "C++", Smalltalk, BASIC, Visual Basic, Pascal, DOS, Windows 95/98 and Windows NT/2K/XP.

Serendipity Systems has a variety of test products available. These include ATE runtime software, fault isolation tools, graphical display utilities and in-circuit test software and hardware.

Serendipity Systems, Inc.
P.O. Box 774507
Steamboat Springs, CO 80477

TEL: (720) 246-8925

www.serendipsys.com

EMAIL: productinfo@serendipsys.com

Index

A

Action Objects 26
Appendix E - Formatting Mask 61
Appendix F - Version Changes 63

C

Comment Window 14
Configuration 28
Configuration Data Dialog 29
Custom Label Dialog 33
Custom Test Types 34

D

Debug Log 15, 50
Display & Behavior 19

E

Edit Menu 8, 44

F

File Definition Dialog 13
File Menu 7, 43
Fill Column Dialog 49
Find Dialog 48

H

Help Menu 12, 47

I

Identification 16

L

Logging & Printing 17

O

Options Menu 11, 46
Other Files 27

P

Power Monitor 35
Prefix Column 40
Prerequisites 23
Project Menu 9

R

Resolving Test Object Links 25

S

Serendipity Systems Inc. 67

Setup Objects 20
Status Bar 5, 42

T

Test Branching 24
Test Criteria 51
Test Files 22
Test Methods 52
Test Object Definition File 56
Test Object Definition Format 57
Test Objects 21
Test Program Definition File 60
TOD Builder 37
TOD Builder Configuration File 53
TOD Identification 58
TOD Verifies 59
Toolbar 42
Toolbars 6
TPD Builder 3
TPD Hiearchy List 4

U

Units Column 41
User Defined 36
User Defined Section 57

V

validate 11, 15, 46, 50
Verify Grid 38
Verify Type Column 39

W

Welcome 1

X

XML Export File 54